

Towards Building a Global Oracle: a Physical Mashup Using Artificial Intelligence Technology

Carolina Fortuna, Matevz Vucnik
Department of Communication Systems,
Jozef Stefan Institute
Jamova 39, Ljubljana, Slovenia
+ 386 1 477 3528

carolina.fortuna@ijs.si, matevz.vucnik@ijs.si

Blaz Fortuna, Klemen Kenda,
Alexandra Moraru, Dunja Mladenec
Artificial Intelligence Laboratory,
Jozef Stefan Institute
Jamova 39, Ljubljana, Slovenia
+ 386 1 477 3528

blaz.fortuna@ijs.si, klemen.kenda@ijs.si,
alexandra.moraru@ijs.si, dunja.mladenec@ijs.si

ABSTRACT

In this paper, we describe Videk - a physical mashup which uses artificial intelligence technology. We make an analogy between human senses and sensors; and between human brain and artificial intelligence technology respectively. This analogy leads to the concept of Global Oracle. We introduce a mashup system which automatically collects data from sensors. The data is processed and stored by SenseStream while the meta-data is fed into ResearchCyc. SenseStream indexes aggregates, performs clustering and learns rules which then it exports as RuleML. ResearchCyc performs logical inference on the meta-data and transliterates logical sentences. The GUI mashes up sensor data with SenseStream output, ResearchCyc output and other external data sources: GoogleMaps, Geonames, Wikipedia and Panoramio.

Categories and Subject Descriptors

H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing – abstracting methods, indexing methods. I.2.0. [Artificial Intelligence]: General. D.2.11 [Software Engineering]: Software Architecture – data abstraction, domain-specific architectures.

General Terms

Algorithms, Performance, Design, Experimentation, Languages.

Keywords

Mashup, artificial intelligence, things, sensors, data mining, machine learning, web services.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WoT 2012, June 2012, Newcastle, UK.

Copyright 2012 ACM 978-1-4503-0624-9/11/06...\$10.00.

1. INTRODUCTION

Humans are born with five senses: sight, hearing, smell, taste and touch. While we grow and develop, we use these senses to observe the surroundings, learn patterns, we learn concepts and associate these concepts with shapes and patterns. Then we are able to generalize, recognize unseen patterns and infer new ones. Now, with the ever increased number of sensors being deployed world-wide, we are developing a global sensing system. A natural next step would be to find a way to connect this system with existing technology such as data mining, machine learning and semantic technologies to develop powerful systems that can help us understand the world – a kind of Global Oracle. The web infrastructure is the ideal means to connect these two worlds: the sensors and the artificial intelligence.

In this paper, we describe Videk [1] - a mashup which makes a small step towards the Global Oracle. The sensor meta-data and measurements are collected and fed into mining, learning and reasoning systems. The output of these systems, the sensor data and external data sources are then mashed into a GUI, but also APIs for external users are provided. The key artificial intelligence technologies powering the system are SenseStream, a storing and processing engine, and ResearchCyc [2], a knowledge base and inference engine derived from the Cyc project.

The contributions of this paper are as follows. We show how the WoT and artificial intelligence technology can be used to create knowledge: having ubiquitous senses spread through the world and based on their input perform learning and reasoning and present the outcome in a useful manner. As such, we introduce and describe SenseStream a stream mining and event detection engine which operated on sensor data. We also use ResearchCyc for modeling the sensor meta-data, reasoning and transliteration of logical statements.

The paper is organized as follows. Section 2 describes the architecture of the Videk mashup system. Section 3 describes the implementation of the internal components of Videk while Section 4 describes the external sources. Related work is described in Section 5 while Section 6 summarizes the paper.

2. SYSTEM ARCHITECTURE

The architecture of Videk is depicted in Figure 1 and consists of four main components: the sensor data, the external sources, the mashup server and the users.

By sensor data we refer to sensor meta-data or context and to sensor measurements. Sensor data is central to the system, as it is relied upon to connect (mashup) with external sources. It can accept sensor data from any source as long as it complies with a specified format. Alternatively, a new adapter for additional sensor data is straightforward to add due to the modular design of the system.

By external sources we refer to any data and/or knowledge source available as a web service. Straightforward examples of these are the Geonames and Wikipedia services which can be queried for information and the response can be parsed and used for the mashup.

The Mashup Server acts as an enabling platform by interfacing with sensor and external data sources as well as with the users. It mashes up raw and processed sensor data with external sources of data and provides a GUI for human end users and an API for external mashups. At the core of the system is the Storing and Processing Engine which transforms the sensor data into useful information and knowledge. The engine stores, indexes and aggregates new sensor measurement data. For instance, it computes statistics such as average, maximum, minimum and it can extract rules and export them as RuleML [3].

By external users we refer to humans and machines. For the human users, a GUI using the mashed up data and knowledge is available. For the machines, the exposed API allows the system to be plugged into other mashups therefore making it another piece in the ecosystem.

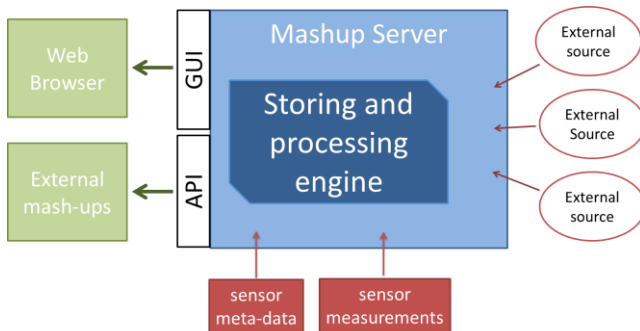


Figure 1 System Architecture

The architecture of Videk is designed in a way that allows organic growth by adding new external sources of data (sensor and external) and updating the user interface to support new sources. The Storage and Processing Engine supports addition of plug-ins to add functionality to the data processing pipeline.

The system is built so that the need for human involvement with respect to data or meta-data provisioning from sensors is reduced to minimum. It achieves this by accommodating automatic collection of sensor meta-data using our custom Device Identification Protocol (DIP). Based on the collected data it performs processing and pulls relevant information from external data sources. The meta-data contains information such as unique

ID of the sensor node, GPS coordinates, configuration (i.e. what type of sensors the node features), unit of measurement for each measurement stream, sampling frequency and accuracy.

All the external services and data sources are interrogated and mashed up starting from the information acquired from the sensors in the form of meta-data. For instance, based on GPS coordinates, the system queries the Geonames service to retrieve the name of the geographical place where the sensor is located, or based on the sensor type (e.g. air pressure), details on how such sensor work or what are expected values can be pulled from Wikipedia.

3. SYSTEM IMPLEMENTATION

Videk is modular therefore it can constantly evolve by adding features and external sources. The current implementation (see Figure 2) uses sensor data collected from VSN/VESNA [5] sensor deployments across Slovenia and Google maps, Geonames, Wikipedia, Panoramio and ResearchCyc as external sources. The Storage and Processing Engine is called SenseStream while the Mashup Server is Apache with some custom Java code, PHP scripts. The GUI uses jQuery for data manipulation, event handling and Ajax interactions.

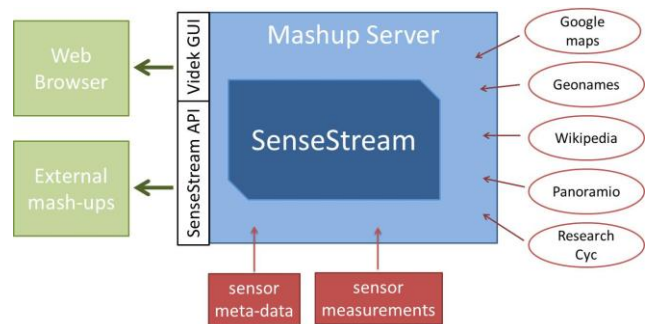


Figure 2 Implementation of the system

3.1 Sensor Data Acquisition

The system is designed to be flexible and accept a variety of data sources. Currently, it uses automatically collected sensor data from VSN/VESNA sensor deployments. The meta-data is automatically collected only once from each sensor node that is joining the network using a custom protocol called Device Identification Protocol. DIP is an application layer protocol which consists of three separate parts. The first part is node discovery, the second is the data collection and the third is the node identification. When a node is discovered, but not identified, the server sends request for identification of that node. A node is discovered once it joins the network of a coordinating node and starts sending measurements to that coordinator. The coordinator then forwards the measurements to the server. If no corresponding metadata is available on the server side, then the server asks for identification. At this point the node will send the meta-data describing itself. The sequence diagram of DIP is depicted in Figure 3.

Currently the meta-data can be collected using a proprietary format or a JSON-LD format. The measurements are sent by the sensor cluster coordinators to the server using HTTP. In the past, the meta-data was inserted by hand. For small deployments, the manual involvement is acceptable while for large deployments the

automated way is preferable hence we developed and implemented DIP. For the time being we prefer DIP over the existing Devices Profile for Web Services (DPWS) [4] which introduces SOA architecture directly on the resource-constrained devices such as sensor nodes. As such, DPWS enables the direct integration of things in the Web service ecosystem by also enabling dynamic discovery. It uses Web Service Definition Language (WSDL) and SOAP/XML messages over HTTP/TCP/IP protocol for meta-data representation while the discovery of web services (WS-Discovery) is done by SOAP over UDP/IP broadcast to minimize the network traffic overhead. DPWS is a relatively mature protocol also from the security point of view. It provides device authentication and message exchange through TLS session.

DIP simplifies this communication by using messages which fit in single link layer IEEE 802.15.4 frame. The payload of a physical frame is 127 bytes saving additional space for higher layer protocol headers. In wireless sensor networks, the radio communication is the most power consuming and the aim was to keep it at minimum assuming the sensor nodes run on batteries. As a tradeoff, DIP does not permit direct integration in the Web service ecosystem. Also, DIP does not provide any security.

For the implementation of DIP we used VSN/VESNA sensor node platform with Contiki¹ operating system. Contiki incorporates a communication stack called RIME which offers features like addressing, broadcast, reliable unicast and reliable bulk unicast for transferring large amounts of data etc. All the mentioned features are needed for the implementation of DIP protocol.

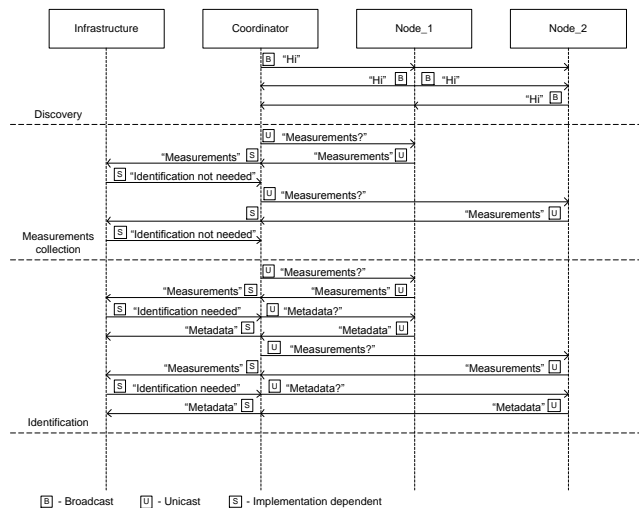


Figure 3 DIP sequence diagram

3.2 SenseStream

SenseStream is a stream mining and event detection engine which powers Videk. SenseStream is built using an internal C++ data mining library and its architecture is depicted in Figure 4. The central part of the system consists of two components: Data Layer and Mining Algorithms [6]. At the bottom of the architecture diagram is a set of data sources, for example sensor

measurements, sensor meta-data, text documents, images. Each data source requires an adapter, which maps the data source to a common interface. Adapters for several standard data sources are provided: a directory of text documents or images, a website, unique users accessing the website, etc.

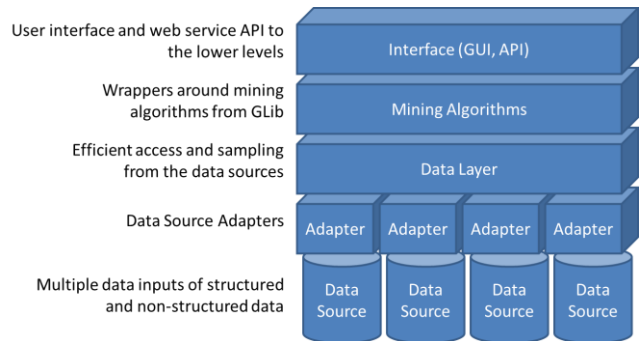


Figure 4 SenseStream architecture

The Data Layer provides unified access to all the data sources from the higher architectural layers, and includes integrated inverted index and multi-modal feature extractors. Integrated inverted index is used to provide faceted search functionality over the records from the data sources. Integrated multi-modal feature extractors provide functionality for extracting feature vectors from raw data provided by the various data sources. Examples of feature extractor would be the vector-space model for text data, visual words for images, or time series statistics for sensor measurements. Feature extractors provide an abstraction layer required by machine learning algorithms.

The measurement feed is processed in real-time. On arrival, each measurement is put in a FIFO queue, and used to update the running aggregate for various time windows. The size of time windows is provided as input parameters. On specified intervals (e.g. minute, hour, day, week), aggregates are stored in a designated store, and indexed using its value, time, period and sensor. The SenseStream API offers browsing, querying and mining of all stored measurements and its aggregates. The results can be exported using Linked Data² standards (RDF, RuleML).

The current implementation of SenseStream uses sensor meta-data and measurements as data sources. On the Data Layer, the implementation the following stores: FIFO queue for measurements (measurement store), aggregates and events (event store), and meta-data (sensor, sensor node and sensor type stores). The current store model is an upgrade of the schema described in [7].

For mining we currently use k-means and hierarchical clustering, association rule mining and basic event detection. SenseStream can export the rules discovered using these data mining algorithms in RuleML format. The RuleML Datalog format provides a simple and clean syntax for expressing “if-then” rules. Each condition is represented by one or more atomic formulas (“Atom”). For example the condition that raindrop exceeds 250 mm per day is represented in Figure 5. The export in the RuleML format is dependent on the vocabulary used for the relation constants (“Rel”). Specialized domain ontologies can simplify the RuleML representation as they can have more specific relations and concepts.

¹ <http://www.contiki-os.org/>

² <http://linkeddata.org/>

```

<And> <Atom>
  <op> <Rel iri="cyc:sensorObservation"/> </op>
  <Var> sensor </Var>
  <Ind iri="cyc:Raindrop"/> </Atom>
<Atom>
  <op> <Rel iri="cyc:doneBy"/> </op>
  <Var> sensor </Var>
  <Var> measurement </Var> </Atom>
<Atom>
  <op> <Rel iri="cyc:measurementResult"/> </op>
  <Var> measurement </Var>
  <Var> vall </Var> </Atom>
<Atom>
  <op> <Rel iri="cyc:duration"/> </op>
  <Var> measurement </Var>
  <Ind type="xs:time">24:00:00</Ind> </Atom>
<Atom>
  <op> <Rel iri="cyc:greaterThan"/> </op>
  <Var> vall </Var>
  <Ind type="xs:float">250</Ind> </Atom> </And>

```

Figure 5 RuleML sample from a rule

3.3 The Videk Mashup Server

The Videk mashup server acts as a glue between different components and services used. It interfaces with sensors receiving data from these, parses and multiplexes the data to the back-up database, to SenseStream, to ResearchCyc, and to the triple store. Then it exposes the API to be used by external applications and a GUI with widgets which mashes up the data and resulting knowledge described above.

3.4 The Videk API and the GUI

The currently exposed API³ allows retrieving sensor meta-data such as list of sensors on a node as well as last measurement from all devices. An example of corresponding calls is:

- <http://hostname/xml/current-state>
- <http://hostname/xml/sensors-on-node/nodeid>

The XML response for the sensors on a given node contains the following information:

- Node
 - id (int) - sensor type id
 - n (int) - number of sensors
 - sensor
 - id (int) - sensor id
 - type (int) - sensor type id

For data manipulation, event handling and Ajax interactions jQuery⁴ library is used. The GUI receives data through the API layer, which is based on a PHP/MySQL custom made content management system (CMS).

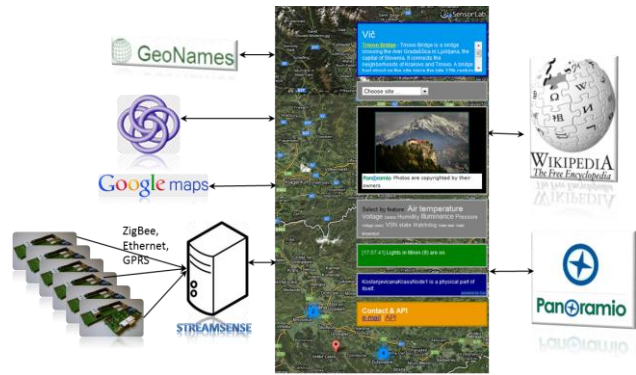


Figure 6 The Videk mashup GUI⁵.

4. EXTERNAL MASHED-UP SOURCES

4.1 ResearchCyc

ResearchCyc⁶ is the research release of Cyc, an artificial intelligence system which is comprised of a knowledge base and a reasoning engine. The idea behind it is to encode knowledge in a structured way and reason about it similar to the way the human mind does it. For instance, each of us learns the concept of a tree, branch, leaf and fruit. Then, we learn relationships between these concepts: that a tree has branches, on a branch grow leaves and fruit. Finally, we are able to recognize instances of these: this apple tree, this apple, this apple leaf. This knowledge builds up in our brains over years and makes it possible to understand, communicate and reason.

In the process of building up such knowledge we use our senses: sight, hearing, smell, taste and touch. Similarly, the purpose of using this technology in the mashup is to allow sensors – the equivalent of senses – to populate the knowledge base – the equivalent of part of our brain – with new knowledge. Finally, as we communicate verbally or using a written form, the Cyc system implements transliteration technology allowing us to generate natural language sentences based on inserted and derived – via logical mechanisms – knowledge.

In the current implementation only sensor meta-data is sent to ResearchCyc. The concepts already exist in the knowledge base and the meta-data represents instances of these concepts. An instance (i.e. individual in Cyc terminology as can be seen in Figure 7 and Figure 8) corresponding to each sensor node and sensor is created. Then the relationships between them together with other data are inserted in the knowledge base. For instance, VicNode1 in Figure 7 is an electronic device which has six instances of sensors connected to it or on-board, for example scp1000VicNode1 and virtualsensor1VicNode1. The TSL2561VicNode1 sensor is an instance of the TSL2561 sensor type and is connected to VicNode1 as stated in Figure 8.

³ <http://sensors.ijs.si/sl/api/index.html>

⁴ <http://jquery.com>

⁵ <http://sensors.ijs.si/>

⁶ <http://research.cyc.com/>

Individual : [VicNode1](#) ¹⁴¹ 

on the term






















isa :  [ElectronicDevice](#)
isa :  [Computer](#)
connectedTo :  [virtualsensor3VicNode1](#)  [TSL2561VicNode1](#)  [sht11VicNode1](#)
(connectedTo [scp1000VicNode1](#) [VicNode1](#))
(connectedTo [virtualsensor2VicNode1](#) [VicNode1](#))
(connectedTo [virtualsensor1VicNode1](#) [VicNode1](#))
(deviceUsed [Testing](#) [VicNode1](#))
hasDevices :  [virtualsensor3VicNode1](#)  [TSL2561VicNode1](#)  [scp1000VicNode1](#)  [sht11VicNode1](#)
(virtualsensor2VicNode1  [virtualsensor2VicNode1](#)  [virtualsensor1VicNode1](#))
latitude :  [@@Degree-UnitOfAngularMeasure](#) 46.042873
longitude :  [@@Degree-UnitOfAngularMeasure](#) 14.487469
nameString : "VicNode1"
objectFoundInLocation :  [IndoorMounting](#)  [Vic](#)
physicalParts :  [virtualsensor3VicNode1](#)  [TSL2561VicNode1](#)  [scp1000VicNode1](#)  [sht11VicNode1](#)
(virtualsensor2VicNode1  [virtualsensor2VicNode1](#)  [virtualsensor1VicNode1](#))
(queryHasVeryHighPertinenceForThing [GetLinkToMap](#) [VicNode1](#))
supportedBy :  [VicBuilding](#)

Figure 7 Knowledge about a sensor node

Individual : [TSL2561VicNode1](#) ¹⁴¹ 

on the term



isa :  [ComputerHardwareComponent](#)  [TSL2561](#)
(connectedTo [VicNode1](#) [TSL2561VicNode1](#))
(hasDevices [VicNode1](#) [TSL2561VicNode1](#))
(physicalParts [VicNode1](#) [TSL2561VicNode1](#))

Figure 8 Knowledge about a sensor attach to a sensor node.

The sensor knowledge is structured in Research Cyc's knowledge base in the form of logical sentences on which inference can be performed. For instance, we only state in the knowledge base that VicNode1 is an ElectronicDevice and the reasoning engine then infers that it's also a Computer. All this knowledge (i.e. asserted and inferred statements) can then be transliterated to natural language as shown for VicNode1 in Figure 9. The ResearchCyc widget in the Mashup GUI displays these statements.


Mt : [BaseKB](#)
(isa [VicNode1](#) [ElectronicDevice](#))
English Translation :
([VicNode1](#) is an [electronic device](#) 

Figure 9 Natural language transliteration of a logical statement.

4.2 External Web Services

The mashup uses external data sources which are available as RESTful web services. Based on the sensor nodes' GPS coordinates, Google maps are used as the GUI's background and the right focus on the deployment locations is presented. Then, also based on the GPS coordinates, the Geonames service is used to retrieve the name of the place where the sensors are deployed. For the deployment on which we zoomed in Figure 6, the name of the neighborhood Vic is displayed together with relevant information from Wikipedia. Finally, the Panoramio service is invoked to retrieve and render pictures of the surrounding area.

5. RELATED WORK

The most relevant body of related work comes from the author of [8] who introduces a four layered architecture for the web of things and illustrates the implementation on three case studies. The architecture is abstracted in four layers which are designed to facilitate mashing up things: accessibility, findability, sharing and

composition. The case studies are a simple, generic sensing application, an energy control and monitoring application for smart houses and an Auto-ID application for supply chain management.

The SensorMap [9] portal and tools allow users to make queries over live data sources and provides mechanisms to archive and index data, process queries, and aggregate and present results on geocentric Web interfaces. The SensorMap architecture has three components: the GeoDB storing sensor metadata, the DataHub handles real time data publishing and the Aggregator which creates icons representing sensor which the users can then mash up with maps.

In [10], Deusto Sentient Graffiti (DSG) context aware mobile mashup for the ubiquitous web is introduced. In DSG, the mobile clients send meta-data and context data for the server. When the mobile client sends a query, then this data is used to return a context aware mashup. The system also uses an inference engine which operates on a knowledge base populated with meta-data. According to a set of rules uses the knowledge to annotate associated to surrounding resources available under their current contextual conditions. Another context based mobile mashup is introduced in [11]. The mobile phone acts as a gateway that collects raw sensor data, preprocesses them and extracts user-centric contexts. The context management platform stores all contexts according to the redefined context ontology, detects the inconsistent information, deduces the high level knowledge using heuristic rules and provides web service application programming interfaces (APIs) to mashup server.

SensorMasher [12] 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.

Our work uses some elements present in the above mentioned systems such as creating a WoT mashup, data indexing and aggregation and overlay on maps, knowledge base and logical inference on meta-data and export of rules and connection to Linked Data. However the work is different from all the above as follows. Our aim is to illustrate how sensors paired with artificial intelligence technology can create a powerful knowledge tool such as a Global Oracle rather than defining a generic architecture for the WoT or for mashups. Our main goal is to illustrate how the WoT technology can be used to create knowledge: having ubiquitous senses spread through the world and based on their input perform learning and reasoning and present the outcome in a useful manner. Finally, we designed SenseStream to become another component in the mashup ecosystem and the overall system to be modular and extensible and impose no limit on the mashed up external sources.

6. SUMMARY AND FURTHER WORK

In this paper, we described Videk - a physical mashup which uses artificial intelligence technology. We made an analogy between human senses and sensors; and between human brain and artificial intelligence technology respectively. This analogy leads to the concept of Global Oracle. We introduced a mashup system which

automatically collects data from sensors. The data is processed and stored by SenseStream while the meta-data is fed into ResearchCyc. SenseStream indexes aggregates, performs clustering and learns rules which then it exports as RuleML. ResearchCyc performs logical inference on the meta-data and transliterates logical sentences. The GUI mashes up sensor data with SenseStream output, ResearchCyc output and other external data sources: GoogleMaps, Geonames, Wikipedia and Panoramio.

As future work we plan to extend SenseStream and couple it with ResearchCyc. With respect to StreamSense, we plan to add additional mining and learning algorithms which makes sense of the input sensor data. Also, we plan to implement detection of events that comply with the learned rules that are currently exported as RuleML. The learned rules, together with richer meta-data automatically collected from upcoming sensor deployments will be also be inserted into ResearchCyc for reasoning and transliteration. All these developments will reflect on the GUI and API in time.

7. ACKNOWLEDGEMENTS

The authors would like to thank Luka Bradesko and Janez Starc for assisting with ResearchCyc and all colleagues from SensorLab for supporting this work. This work was in part supported by the Slovenian Research Agency (ARRS) and the ICT Programme of the EC under ENVISION (ICT-2009-249120) and PlanetData (ICT-NoE-257641) and the competence center OPCOMM.

8. REFERENCES

- [1] Kenda, K., Fortuna, C., Fortuna, B., & Grobelnik, M. Videk: A Mash-up For Environmental Intelligence. In Proceedings of the 8th Extended Semantic Web Conference (ESWC 2011) (Heraklion, Greece, May 2011).
- [2] Lefkowitz, L.; Curtis, J., Witbrock, M., 2007. Accessible Research Cyc, Final technical rept. Dec 2003-May 2007.
- [3] Boley, H., Tabet, S., Wagner, G. 2001. Design Rationale of RuleML: A Markup Language for Semantic Web Rules. In proceedings of the first Semantic Web Working Symposium (Stanford University, California, USA, July 30 - August 1, 2001). SWWS'01.
- [4] Jammes, F.; Mensch, A.; Smit, H.; Service-Oriented Device Communications Using the Devices Profile for Web services, Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on , vol.1, no., pp.947-955, 21-23 May 2007
- [5] Smolnikar, M., Platise, U. 2010. Versatile Sensor Node - A Platform for the Sensor as a Service Concept. In Proceedings of Prosense WSN and SME workshop: Wireless sensor networks (WSN) and small medium enterprises (SME) (Ljubljana, Slovenia, May 20, 2010).
- [6] Fortuna, Blaž. Semi-automatic ontology construction: doctoral dissertation = Polavtomatska gradnja ontologij: doktorska disertacija. (Ljubljana, Slovenia, October 2011).
- [7] Moraru, A., Vučnik, M., Porcius, M., Fortuna, C., Mohorčič, M., Mladenčić, D. 2011. Exposing Real World Information for the Web of Things. In Proceedings of Eight International Workshop on Information Integration on the Web in conjunction with WWW'11 (Hyderabad, India, March 28, 2011). IJWeb2011. DOI=<http://doi.acm.org/10.1145/1982624.1982630>.
- [8] Guinard, D. 2011. A Web of Things Application Architecture – Integrating the Real-World into the Web. PhD thesis No. 19891, ETH Zurich, Zurich, Switzerland
- [9] Nath, S., Liu, J., Zhao, F. 2007. SensorMap for Wide-Area Sensor Webs, IEEE Computer, Vol. 40, Issue 7. July 2007. DOI=<http://doi.acm.org/10.1109/MC.2007.250>
- [10] López-de-Ipiña, D, Vazquez, I., Abaitua, J. 2007. A Context-Aware Mobile Mashup for Ubiquitous Web. In Proceedings of IET International Conference on Intelligent Environments (University of Ulm, Ulm, Germany, Sept. 24-25, 2007). IE2007.
- [11] Li, Y., Fang, J., Xiong, J. 2008. A Context-Aware Services Mashup System. In Proceedings of 7th International Conference on Grid and Cooperative Computing (Shenzen, China, Oct. 24-26, 2008). GCC'08. DOI=<http://dx.doi.org/10.1109/GCC.2008.62>.
- [12] Le-Phuoc, D., Hauswirth, M. 2009. Linked open data in sensor data mashups. In Proceedings of the 2nd International Workshop on Semantic Sensor Networks (Washington DC, USA, Oct. 25-29, 2009). SSN'09.