Design of a Semantic and Holistic Architecture for Sensor Data Integration

Aitor Corchero Rodriguez

University of Lleida

Author Note

Aitor Corchero is a computer engineer by the University of Mondragon. Currently, Aitor is a researcher in the Environment and Food department of BDigital Technologic Centre. Mainly, Aitor's work is focused on addressing intelligent tools aligned with the water, energy and food management projects.

“Knowing your own ignorance is the first step to enlightenment.”

Patrick Rothfuss (Wise Man’s Fear)

ACKNOWLEDGEMENTS

During the Msc Thesis, I am very grateful for the given support in both, the University of Lleida and BDigital Technology Centre (BDigital) as well as my friends and family.

My sincere and deep gratitude to my advisor, Phd. Roberto Garcia Gonzalez (*Master Lorren*), for the guidance and consistent advices. Furthermore, I would like to say thanks to other teachers of the University of Lleida as *Master Kilvin* for the long afternoons melding and burning the future ever-burning lamps. Also, I would appreciate the kindly support of *Master Arwyl* for the discussions and deeply conversations. Special mention is needed for *Master Elodin* that in spite of not being a full-time professor at University, is one of the craziest man in the universe and also he has helped me in understanding the meaning of "*The name of the Wind*". Moreover, I would like to appreciate my work colleagues, *Simmon*, *Willem*, *Manet* and *Mola* because without them, the nights in "*The Eolian*" wouldn't be the same. Furthermore, I would like to appreciate the unlimited patience of *Mandrag* and *Arthur Herma* for the advices and knowledge that has taught to me.

I would particularly like to say thank to my family. Always supportive and encouraging, *Arliden* and *Laurian* that are my foundation. I know I owe them my life. I appreciate the kindly love of my sister, *Fela,* and *my brother in law* that also are one of my best friends. In reference to my friends, I only have gratitude to *Devan Lochees* for writing the best chapters of my life, *Bast* for being one of the main character in my story and the mentioned *Master Elodin*. I don't forget *Abenthy* because you started me in the Artificial Intelligence world as my previous advisor. Additionally, I would appreciate the unconditional support of *Auri* that you taught things like: "*Owls are wise. They are careful and patient. Wisdom precludes boldness. That is why owls make poor heroes*". Finally, I don't forget the woman who spend with me the days of the life, *Denna*

because “*you are the notes of my music and the queen of my heart*”. So, my name is “*Kvothe, you may have heard of me”*.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

GLOSSARY OF TERMS

AMQP	- Advanced Message Queuing Protocol
API	- Application Programming Interface
CAN	- Controller Area Network
CBRN	- Chemical, Biological, Radiological and Nuclear
CCSI	- Common CBRN Sensor Interface
CEP	- Complex Event Processing
CityML	- City Markup Language
CSV	- Comma Separated Value
D2RQ	-Data to Relational Query
DCS	-Distributed Control System
DHS	- Department of Homeland Security
DHS-ANSI-N42.XX	- Department of Homeland Security-American National Standard Institute-N42.XX
DKXML	- Device Kit Markup Language
DMS	- Data Model for Sensor
DoD	- Department of Defense
DTRA	-Defense Threat Reduction Agency
EDDL	- Electronic Device Description Language
EDDL	- Electronic Device Description Language
ESB	- Enterprise Service Bus
Geo-SPARQL	- Geospatial SPARQL
GHG	- GreenHouse Gas
GIG	- Global Information Grid
GIS	- Geographic Information Systems
GML	- Geospatial Markup Language
GSN	- Global Sensor Networks
HaaS	- Humans as a Sensors
IEC	- International Electro-technical Commission
IEEE1451	- Institute of Electrical and Electronics Engineers 1451
IoT	- Internet of Things
ISO	- International Organization for Standardization
JPEO-CBD	- Joint Program Executive Office for Chemical and Biological Defense
JSON	- JavaScript Object Notation
JSON-LD	- JavaScript Object Notation for Linked Data

JWARN- Joint Warning and Reporting Network

LODC- Linked Open Data Cloud

MIME-Multipurpose Internet Mail Extensions

MMI- MultiModal Architecture

NCAP- Network-Capable Application

NECES-Net-Centric Enterprise Services

NIST- National Institute of Standards and Technology

O&M- Observation and Measurement model

OGC-Open Geospatial Consortium

OGC-SAS-OGC- Sensor Alert Service

OGC-SOS- OGC- Sensor Observation Service

OGC-SPS -OGC-Sensor Planning Service

OGC-WMS- OGC- Web Map Services

OGC-WNS- OGC-Web Notification Service

OLE- Object Linking and Embedding

OPC- OLE for Process Control

OSGI- Open Services Gateway Initiative

OWL- Web Ontology Language

QoS- Quality of a Sensor

R2RML- RDB to RDF Mapping Language

RDF- Resource Description Framework

RDFa- Resource Description Framework in attributes

REST- REpresentational State Transfer

RFID- Radio Frequency IDentification

RIF- Rule Interexchange Format

SensorML- Sensor Markup Language

SOA- Service Oriented Architecture

SOAP- Simple Object Access Protocol

SODA- Service Oriented Device Architecture

SoSCOE- System of Systems Common Operating Environment

SPARQL- SPARQL Protocol and RDF Query Language

SPIN- SPARQL Inferencing Notation

SQL-Structured Query Language

SRSD- Semantic Repository of Sensor Data

SSN-XG- Semantic Sensor Network Incubator Group

SSW- Semantic Sensor Web

SWE- Sensor Web Enablement

SWEET- Semantic Web for Earth and Environmental Terminology

SWRL- Semantic Web Rule Language

TBI- Transducer Bus Interface

TCP/IP-Transmission Control Protocol/Internet Protocol

TEDS- Transducer Electronic Data Sheet

TII- Transducer Independent Interface

TransducerML- Transducer Markup Language

UPM-Polytechnic University of Madrid

VGI- Volunteered Grafical Inforamtion

W3C- World Wide Web Consourtium

WaterML- Water Markup Language

WKT- Well Known Text

WPAN - Wireless Personal Area Network

WSDL- Web Services Description Language

WTIM- Wireless Transducer Interface Module

XML-eXtensible Markup Language

ABSTRACT

To date, Semantic Sensor Web research and development has focused on establishing common techniques and practices that homogenize how to discover sensors, collect their data, integrate them, extract information from them, etc. However, as these issues are overcame and huge data bases of sensor data begin to emerge, the focus should change to improve the data management and the information overload, discarding the non-relevant information from the relevant one, and on the other hand, allow easy and intuitive navigation through it. The objective is to move up the wisdom hierarchy and empower users so they can start discovering new relevant knowledge and making decisions based on that. Therefore, the present research depicts the development of an early-prototyped architecture that emphases the usage of streaming semantic technology to collect, process and converting sensor information in data events that, with the application of semantic reasoning, the systems will automatically and autonomously understand sensor information and also will be able to detect simple situation awareness. Finally, the work performed under this research is partially founded by "*FP7 EU WatERP*" project (*GA318603*).

Keywords: *sensor, data, Semantic Web, decision support, exploration, visualisation, stream reasoning.*

INTRODUCTION

During last few decades, we have witnessed the rise of sensors and applications based on them has become an integral part of our lives. Nowadays, sensors are connected between them using the Internet (and the Web), where sensing data is transferred. Based on *(Cory Henson, Thirunarayan, & Sheth, 2011)*, millions of sensors are used around the globe generating avalanches of data that feed applications related with different domains such as education, environment or security to mention a few. Regarding data generated, in *(Cory Henson, Sheth, & Thirunarayan, 2012)* is remarked that "*data generated and shared on the Web by sensors during this last years is more than the generated in the past 40,000 years*". This last aspect is also reinforced by the rise in the number of sensors during the last years, achieving an estimated amount of *40* billion of sensors connected between them, mobile devices and computerised systems *(Nokia, 2008)*.

Therefore, there is an enormous value in being able to manage sensors over the Web in order to provide mechanisms to collect, manage and share sensor information and knowledge. For instance, by generating and identifying those non-desirable situations whose early detection can save lives (e.g. early detection of water floods), reduce damage (earthquakes detection) and/or detect inefficient energetic situations that avoid to reduce GreenHouse Gas emissions (GHG).

Based on these necessities, current sensor trends and data sensing managing has skewed to **Semantic Sensor Web** concept (SSW)*(Sheth & Henson, 2012)*. Under this concept, sensor and sensor network information is connected semantically and then, sensor related information is fed with necessary meaning to provide cross-domain usage that help in the detection of the non-desirable situations during the generation of early warning systems.

Based on this last concept, the present thesis *(Corchero, Domingo, & García, 2013)* is situated under the usage of **SSW** concept and the existing architectures in order to enrich standard architectures with mechanisms that support decision support empowering users with data exploration and visualisation tools. Then, the proposed knowledge discovery environment is based on an architecture that integrates existing tools for sensor data collection and integration. So, mentioned approach is then extended with data exploration and visualisation mechanisms that permit to *(i)* harmonise sensor information into a single platform; *(ii)* pre-process sensor information in order to create "*events*" that represent a certain situation based on sensor observations at specific time; *(iii)* visualise the information based on the geo-graphical zone where the sensor actuates; and *(iv)* gather user information and knowledge that is semantically included into the collected information in order to learn about user experience and apply this experience in further sensor collections.

Based on this initial approach, the present thesis is structured as follows: The first section, "*INTRODUCTION*" describes the framework of the thesis where the sensor concept is linked up with main approaches able to collect and share sensor data throughout the web (sensor web concept). In spite of data management issues, this concept is trending on the road to approaches that provide meaning to sensory information, **semantic sensor web** concept. Follow up the introduction, the "*BEYOND THE STATE OF THE ART*" section depicts the current sensor web evolvements towards standardising the process of data collection and sharing. Furthermore, this section also describes the current maturity of semantic sensor approaches going with the identification of main challenges corresponding to this approach. Based on the identified challenges, the "*OBJECTIVES*" section shows the main aims of the present research. In order to achieve this objectives, in the "*DESIGN OF THE ARCHITECTURE*" section, general semantic sensor web architecture that accomplish most of the challenges is described. Establishing the architectural design as a starting point, the "*EARLY DEVELOPMENT OF THE ARCHITECTURE*" describes the development state of designed architecture by including also initial results over the architecture. At last but not least, the "*CONCLUSIONS AND FUTURE WORK*" represent the main outcomes and future work to be accomplished in relation with this research.

Sensor and Sensor Network concept and motivation

Sensors are materials or devices which change their (conductive) properties according to a physical stimulus *(Moraru & Mladenić, 2012)*. From an engineering point of view, sensor/s are defined as a device/s that convert a physical, chemical, or biological parameter into an electrical signal *(Arne Bröring et al., 2011)*. Then, this electrical signal serves to measure certain situation that the sensor observes. This electrical signal is called "*sensor Observation*", "*sensor Result*", or "*sensor Output*". The generated output from the sensor can be sent to other sensors and/or systems by establishing a specific communication. In both cases, the communication between sensors and/or other systems can be done over a wired or wireless networks by establishing a specific communication language that can be private or standard depending of the sensor. Then, by using a communication protocol, sensor can be organized or connected with other systems in a coordinate manner. When this situation occurs, sensors are structured in a **sensor system** and/or **architecture**. Therefore, a sensor system is an aggregation of sensors spatially distributed that are attached into a single platform by using a common communication language (or protocol).

Independently of sensor or the sensor architecture, this systems can be shown as a resources that provide certain information to other systems (data bases, other sensor architectures) across application locally situated (technology-driven approach) or spatially distributed (web-based approach).

The technology-driven approach is based on developing interfaces to communicate applications/systems with sensor/s directly or using a common sensor interface based on distributor specifications. This approach is exposed to frequent changes in data retrieval and data structures, since new platforms emerge, old ones disappear, and prevailing ones modify their user and programming interfaces. Then, this approaches are strongly dependent of sensor interfaces generating highly adaptation costs when sensor architecture changes.

For that reason, sensor systems has evolved to web-based approach that is focused on a combination of sensor networks with the web. This current approach is called **Sensor Web** or **Sensor Internet**. This concept was emerged in *1997* with a large scale sensor architecture where autonomous sensor nodes act and coordinate as a whole performing standalone observations and/or cooperative tasks *(Pileggi, 2010)*. Originally, sensor web terms refers to a wireless sensor network architecture in which the nodes of the network were autonomous and able to react to the data measured by themselves and other nodes in the network. However, this term has been evolved towards the current used definition that includes an open-informational sharing across platforms and sensors around the globe as mentioned in *(Sigüenza, Díaz-Pardo, & Bernat, 2012)*: "*Sensor Web reflects such a kind of infrastructure for sharing, finding, and accessing sensors and their data across different applications. It hides the heterogeneous sensor hardware and communication protocols from the applications built on top of it*". So, current sensor web concept assures interoperability and universal access to sensor observations and measurements. Then, integration of heterogeneous data can be seen from different point of views: *(i)* *Syntactic*, to overcome technical heterogeneity; *(ii)* *Semantic*, to overcome ambiguities and different interpretations; *(iii)* *Application*, to deliver sustainable and re-usable concepts and components of the application domain; and *(iv)* *Phenomena observation*, to evaluate the meaning of the observation from temporal, spatial and thematic perspectives.

Main efforts in this area are focused on the provision of platforms that could be used to build sensor-web based applications more efficiently, considering some of the most important challenges in sensor-based data management and sensor network configuration *(Arne Bröring et al., 2011)*. Then, an emerging number of Sensor Web portals, such as Sensorpedia[1], SensorMap[2], or Xively[3], are currently being developed to enable users to upload and share sensor data. Additionally, virtual sensor web portals are also being emerged in order to simulate sensors based on XML and/or CSV files. In this ambit, portals and libraries such as Esper[4] and Global Sensor Networks (GSN) library *(Aberer, Hauswirth, & Salehi, 2006)* permit to treat sensor information as an events in a virtualized sensor network.

Furthermore, the sensor web improves the Internet of Things (IoT) concept towards providing huge benefits to the society and business. The IoT concept still lacks many technological issues which needs to be addressed. First of all, IoT does not provide any registry mechanism for publishing service information publicly that is hosted on sensor. Secondly, different sensors and devices comprise with different data formats and models, thus causing IoT to exhibit deficiency in discovering and composing diversified services. Thirdly, IoT deficient in handling service invocation that sensor triggers with the occurrence of an event. Then, the interaction between events and services are absent in current IoT clouds. Moreover, authorized

[1] Sensorpedia Web page: http://www.sensorpedia.com/
[2] SensorMap Web page: http://www.sensofar.com/sensofar/products/sensomap.html
[3] Xively Web page: https://xively.com/
[4] Esper Web page: http://esper.codehaus.org/tutorials/tutorial/presentations.html

access to IoT cloud sensor data and services without relaxing user privacy is also a challenging task.

Semantic Sensor Web concept and motivation

In spite of sensor web concept provides fruitfully advantages for sharing and making accessible data, the main weakness of this concept resides in the collection of huge amount of data without generating effective knowledge. In words of *(Sheth & Henson, 2012)*: "*The lack of integration and communication between these (sensor web) networks, however, often isolates important data streams and intensifies the existing problem of too much data and not enough knowledge*".

Then, a new approach in which sensor information and data is enhanced with semantic layer, has been emerged. Hence, the basis of this approach be inherent in the semantics or information meaning that is formally defined in form of metadata with the main aim of increasing interoperability at same time as contextual information that is essential for establishing situational knowledge, is also providing. This concept was named **Semantic Sensor Web** and was formally defined by *(Sheth & Perry, 2008)* in 2008 as "*a framework for providing enhanced meaning for sensor observations so as to enable situation awareness. It enhances meaning by adding semantic annotations to existing standard sensor languages of the Open Geospatial Consortium. Sensor Web Enablement (OGC-SWE)*".

That means, the **semantic sensor web concept** is based on enriching eXtensible Markup Language (XML)-based sensor communication language (e.g. OGC-SWE) with metadata standard information from semantics to automate the process of *(i)* registering sensor description into the sensor web service architecture; *(ii)* converting native sensor protocol to higher level sensor web observations; *(iii)* matching between sensor characteristics and service model in order to correctly upload sensor observations; and *(iv)* adding spatial and temporal information to the sensor observations. This semantic enhancement is done by applying Resource Description Framework (RDF) and/or Web Ontology Language (OWL) vocabularies in terms of adding RDF/OWL annotations. Furthermore, the semantic measurement can be measured in terms of setting semantic rules and restrictions over the existing syntactical meta-models.

Current motivation of **semantic sensor web** concept is focused on **interconnecting identifiable things (e.g. devices, people, applications, services, etc.) related with sensor information and location to better understand their surrounding environment**. The understand of a situation or context support the generation of intelligent decisions to respond to the dynamics of the environment *(Barnaghi & Wang, 2012)*. For that, visualization and semantic knowledge transportation as RDF in attributes (RDFa) *(Herman, Adida, Sporny, & Birbeck, 2013)* and JavaScript Object Notation for Linked Data (JSON-LD) *(Sporny, Longley, Kellogg, Lanthaler, & Lindström, 2014)* are being included into the growing architectures. In reference to ontologies description, service domain ontologies (e.g. OWL-S[5]) are being also added into the domain description (sensor environment) in order to understand and automate the sensor registering process *(Arne Bröring et al., 2011)*. In reference to ontological rule representation, Rule Interexchange Format (RIF) *(Kifer & Boley, 2013)* and SPARQL Protocol and RDF Query Language (SPARQL) Inference Notation (SPIN)[6] rules are being used against Semantic Web Rule Language (SWRL) *(Horrocks et al., 2004)* rules thanks to the first approaches facilitates rules interexchange based on SPARQL syntax by the usage of Construct clause.

[5] OWL-S Web site: http://www.w3.org/Submission/OWL-S/
[6] SPIN Web Page: http://spinrdf.org/

Finally, additional challenges include integration and fusion of data from different sources, describing the objects and events, data aggregation and fusion rules, defining thresholds, real-time processing of the data streams in large scale, and quality and dynamicity issues *(A Bröring, Janowicz, Stasch, & Kuhn, 2009; Phuoc, Parreira, & Hauswirth, 2010)*.

BEYOND THE STATE OF THE ART

As depicted in "*Semantic Sensor Web concept and motivation*" section, **semantic sensor web** is a progressive concept in which main core is based on the semantics and the benefit that they offer in relation with semantic interoperability between systems/architectures, data harmonization from different data sources (e.g. web data, database information, syntactic information, etc.), direct data integration and sharing with Webs, applications and architectures by using semantic data exchange languages (e.g. RDFa or JSON-LD), and direct data and event generation by the use of rules based on the SPARQL syntax (e.g. SPIN or RIF rules).

Based on this concept and the benefits they offer, the present section is aimed at reviewing the main challenges of this technology in order to establish the starting point of current research. For that reason, current review has been focused on analysing current sensor languages and architectures (see Section "*State of the art in sensor languages and data exchange*") in order to catalogue syntactic information (XML, Comma Separated Value -CSV- and other formats) that sensor uses to interexchange data and it inherent information.

Once sensor languages and data exchange has been analysed in terms of sensor representation and data sharing, main challenges regarding semantic sensor web has been identified (see Section "*State of the art in semantic sensor web*"). This challenges depict the current and future research lines in order to make more interoperable and open the **semantic sensor web** concept towards detecting undesirable situations and providing data sharing across different domains.

State of the art in sensor languages and data exchange

Nowadays, the information and data generation by the computerised systems grows up day by day. There exist many standards, emerging standards or standards that are being developed in sensor areas. These standards has been designed to serve the users perspectives. However, current standards have some degrees of overlapping and are designed to provide to the user huge amount of information without not enough meaning. Complemented to the not enough knowledge generation from data, current standards also are not sufficient defined in reference to resource reallocation and resource sharing. This last aspect is aligned with the non-existence of sufficient multi-layered or cross domain standards definition. Consequently, existent standards are not enough collaborative with other standards in spite of the same domain actuation. This main aspect generates too much difficulties in order to generate automatic solution for a specific domain. Therefore, there exist a lack of uniform operations and standard representation for sensor data in most of sensor domains. Each industrial fabricant of sensors impose it sensor protocol, data exchange and other important capabilities that makes difficult to interoperate with the rest of the sensor environment.

Hence, sensor data and information has to be exchanged by different user and domains in order to enhance data comprehension and then, improve for example, alerting systems. The best way to accommodate this situation is to harmonise these standards towards achieving the highest degree of interoperability *(Chen & Helal, 2008)*. The benefits that can provide the standards harmonisation are aligned with the promotion of measurement, standards, and technology to enhance productivity, facilitate trade, and improve the quality of life. Furthermore, this standards languages has to facilitate the interoperability between domains and devices. Therefore, the interoperability has to derive towards an open and participatory environment while encouraging all actors to take responsibility with their collective mitigations and adaptations to change. Additionally, these open-standards may take into account the sensors and devices that are usually tightly coupled with the specific location, application, and environment where actuates.

For these reasons, last research and trends in sensor standards rely on the definition of a common open architecture to gather information uncouple for the sensor language from the specific domain *(Lee, 2007)*. This is the case of the Institute of Electrical and Electronics Engineers 1451 (IEEE1451) *(IEEE, 1999)*, OASIS- Emergency Data Exchange Language (EDDL)*(Raymond, Webb, & Aymond, 2006)*, Electronic Device Description Language (EDDL) *(EDDL, 2007)*, Device Kit *(Kit, 2013)* or OGC-SWE *(OGC-SWE, 2011)* that have focused the efforts on generating a common sensor language that serves multiple environments meanwhile offers the information in a common, standard and understandable machine-readability language (e.g. XML). Other research lines has been less ambitious and has been focused on generating a common language in a certain environment. This is the case of Chemical, Biological, Radiological and Nuclear (CBRN)-Sensor Interface *(JPEO-CBD & SSA, 2008)*, Department of Homeland Security-American National Standard Institute-N42.XX (DHS-ANSI-N42.XX) *(NIST, 2006)* or Electronic Device Description Language (EDDL) (Raymond et al., 2006) that has been motivated by improving of homeland security.

As a reality, future trends in sensor languages and standards will rely on the generation of an open standards that includes all relative sensor information, data provenance and geographical information in order to facilitate data understanding to improve computerised systems. Furthermore, future trends in sensor language and architectures will be concentrated on generating knowledge and wisdom from the huge amount of data collected.

Sensor standards

This section is aimed at presenting most relevant sensor standards languages and architectures in order to determine current trends based on most used standards that permit to model a common vocabulary to share, extract knowledge and enhance the interoperability between systems that consumes and needs sensor information.

Most representative standards and its main capabilities has been described on the *Table 1*. This table has been constructed based on the previous work performed by *(Chen & Helal, 2008; Hu, Robinson, & Indulska, 2007; Lee, 2007)* where some sensor languages and metrics for classifying this languages were defined. So, in this table is described the domains where the standards actuates. As seen, multiple-domains contains different standards such as the homeland security. However, standards such as OGC-SWE, IEEE1451 and Device Kit has been modelled by focusing on sensor and actuator nature decoupling sensor data collection and actuation from the domain.

In reference to the architecture, some of the standard languages are a small part of an upper architecture. This upper architecture is defined as Service-Oriented Architecture (SOA) or Service Oriented Device Architecture (SODA) in most of the cases. The SOA is a design pattern or concept enabling the applications to be connected by using services or service operations. Then, this kind of architecture can be based on Simple Object Access Protocol (SOAP) services or REpresentational State Transfer (REST) services depending of the needs to consume data. Similarly, the SODA concept permits the devices to be connected by using web service. In this case, the architecture is based on an Enterprise Service Bus (ESB) in charge of managing the devices and offering the service to the elements connected to the ESB. Furthermore, more simplistic approaches rely on the possibility of using an Application Programming Interface (API) model to interconnect the devices with systems. Therefore, the standards architecture is divided into an Object Oriented and/or Data Oriented architectures.

Regarding with the standard communication language or encoding, old standards were focused on a plain text or commands as a main mechanism for intercommunicate sensors with other systems. In contrast, current standards are focused on XML language as a main mechanism to exchange information. The XML facilitates the sensor information understanding meanwhile gives sense to the measured information, sensor information and/or sensor configuration. Henceforth, the most current standards uses are able to represent complex data types in contrast with older standards that are focused on simples ones.

As an impact on standards characterisation, the software implementation of these models have been mainly closed environments and or limited ones. This non wide-acceptance of the standards can provoke not enough interoperability between standards and a strongly dependence with the sensor fabricants and industry. As a broken line, the OGC-SWE offers and open and easy-extensible architecture that generalise the sensor intercommunication and informational gathering. This architecture are implemented by the "*52º North*"[7] that provides to the user the needed guides and best practices to deploy the OGC-SWE architecture.

[7] 52º North Initiative Web page: http://52north.org/

	ECHONET	**IEEE 1451.X**	**OGC-SWE**	**CBRN Sensor Interface**	**DHS-ANSI N42.XX**	**OASIS-EDDL**	**Device Kit**	**EDDL**
Domain	*Home appliance*	*Transversal to communicate intelligent sensors*	*Transversal to communicate intelligent sensors*	*Security and Safety*	*Radiation Measurement in homeland security applications*	*Homeland Security*	*Sensors and Actuators*	*Sensor and Devices Communication*
Architecture	*Bus architecture*	*Bus Architecture*	*SOA-Architecture*	*Net-Centric Platform*	*NA*	*DCS architecture*	*SODA based on OSGi Model*	*SOA-Architecture*
Communication Language	*Commands (events) in plain text*	*Commands (sensor communication) and IDL on sensor configuration (XML-based)*	*XML*	*API with XML schema*	*XML*	*XML*	*API with XML schema*	*Plain Text*
Design Perspective	*Object-Oriented*	*Modular by sensor type*	*Data oriented*	*Object Oriented*	*Data Oriented*	*Data Oriented*	*Modular*	*Data Oriented*
Device Model	*Single-object*	*Multiple Blocks on TEDS*	*Observation process*	*Single Object*	*Radiation instruments data measurements*	*Alerts Information Measurement*	*Multiple Layers*	*Single Device-cross layer*
Basic Component	*Device*	*Block*	*Observation process*	*Device/System*	*Radiation Instrument*	*Alerts and Resource Messages*	*Device Agent*	*Device*
Composite Component	*NA*	*Device*	*Observation process and sensor information*	*NA*	*Radiation Instrument information and time series values*	*Alerts and Resource Messages related information*	*Device Agent*	*Device*
Measurement Modelling	*Basic/primitive Data Types*	*Complex Data Types (sensor Networks)*	*Complex Data Types (aligned with standards)*	*Complex Data Types*	*Complex Data Types*	*Complex Data Types*	*Basic/Primitive Data Types*	*Basic/Primitive Data Types*
Software Support	*System Component for ECHONET (Yaskawa Information Systems Corp.)*	*NA*	*52º North*	*JWARN, JEM and JOEF-*	*NA*	*NA*	*Device Kit SODA architecture*	*EDDL Tool*

Table 1 “Sensor languages harmonization”"

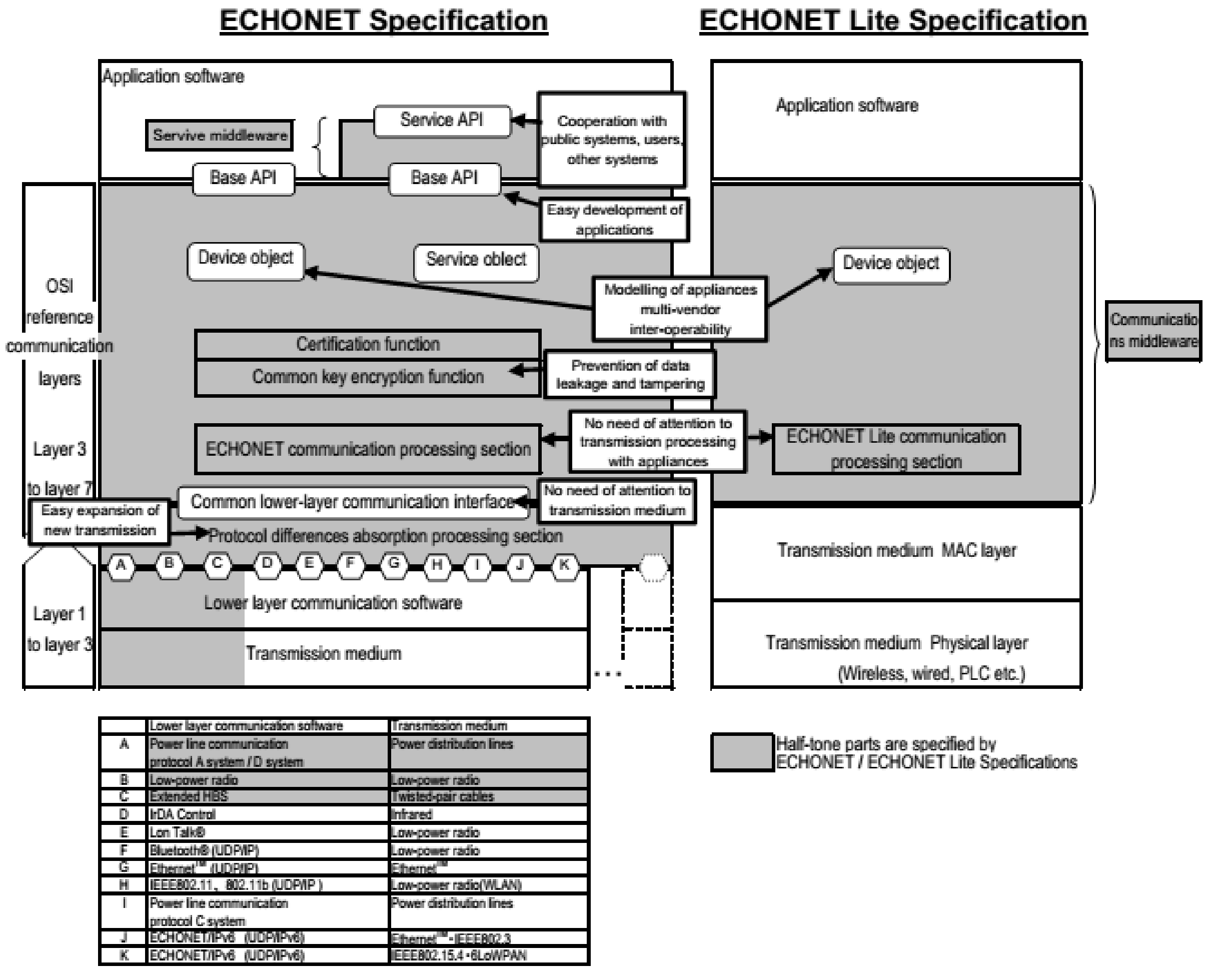
ECHNET

Energy Conservation and Homecare Network (ECHONET) is a standard developed in Japan in *1997 (ECHONET, 2013)*. The main purpose of this standard is to promote the sustainable development of the residential and household environment. Thus, the ECHONET standard aim is to reduce CO_2 emissions, reduce healthcare cost and build safe and barrier free societies. Since a technical point of view, the ECHONET standard language provides a mechanism to include under the same architecture different sensors, transducer and home appliance in order to monitor energy, control the home parameters and improve residential safety. Then, the ECHONET is a household systematic approach that integrates many devices and systems.

The mentioned standard language features are focusing on *(i)* applying for ordinary households, multi-dwelling buildings, shops, and small- and medium-sized office buildings; *(ii)* allowing use of detailed rules (ECHONET device objects) for all types of household equipment (*87* categories)- including household devices, residential appliances, sensors, and health management devices- as ECHONET devices; *(iii)* allowing easy installation and changing of systems through plug & play functions; *(iv)* permitting free combination of transmission media (e.g. wireless media, wired media, etc.) according to the installation environment; *(v)* making product cost reduction easier by employing a middleware adaptor system; and *(vi)* international standardisation in International Organization for Standardization (ISO) and International Electro-technical Commission (IEC).

The ECHONET architecture is formed by *(i)* physical architecture in charge of communicating with the installed physical elements of the household; *(ii)* Communication layer aimed at abstracting and exchanging information between different devices; and *(iii)* service layer that has the objective of providing needed mechanisms to cooperate with external sources and users.

Moreover, the ECHONET specification (see *Figure 1*) has been subdivided into the lite and the complete specification (full standard). The difference between the two specifications lies on the number of devices that permit to connect, type of services that are able to communicate and the physical layer transmission protocols.


ECHONET Specification
ECHONET Lite Specification
Application software
Servive middleware
Service API
Cooperation with public systems, users, other systems
Base API
Base API
Easy development of applications
Device object
Service oblect
Modelling of appliances multi-vendor inter-operability
Device object
OSI reference communication layers
Certification function
Common key encryption function
Prevention of data leakage and tampering
Layer 3 to layer 7
ECHONET communication processing section
No need of attention to transmission processing with appliances
ECHONET Lite communication processing section
Common lower-layer communication interface
No need of attention to transmission medium
Easy expansion of new transmission
Protocol differences absorption processing section
A B C D E F G H I J K
Layer 1 to layer 3
Lower layer communication software
Transmission medium
Application software
Communications middleware
Transmission medium MAC layer
Transmission medium Physical layer (Wireless, wired, PLC etc.)
Half-tone parts are specified by ECHONET / ECHONET Lite Specifications

	Lower layer communication software	Transmission medium
A	Power line communication protocol A system / D system	Power distribution lines
B	Low-power radio	Low-power radio
C	Extended HBS	Twisted-pair cables
D	IrDA Control	Infrared
E	Lon Talk®	Low-power radio
F	Bluetooth® (UDP/IP)	Low-power radio
G	Ethernet™ (UDP/IP)	Ethernet™
H	IEEE802.11、802.11b (UDP/IP)	Low-power radio(WLAN)
I	Power line communication protocol C system	Power distribution lines
J	ECHONET/IPv6 (UDP/IPv6)	Ethernet™ - IEEE802.3
K	ECHONET/IPv6 (UDP/IPv6)	IEEE802.15.4 • 6LoWPAN

Figure 1 "ECHONET architecture". Source: (ECHONET, 2013)

Based on this architecture, the ECHONET standard protocol to communicate sensors are object oriented modelling of system configuration. That means, the architecture is based on a SODA where devices (elements) are capable of communicating with the rest of appliance using like-an Open Services Gateway Initiative (OSGI) protocol. Furthermore, ECHONET permit to connect different domains between them by using gateways able to flow the data between domains.

Furthermore, the categorisation of the architecture is open and then, household vendors and clients can download the specification and implement this communication language. Regarding to the sensor discovering, the ECHONET permit the device to be plug and play. That means, when you connect a compatible device into the network, the systems automatically detect and establish a communication with the installed sensor.

Additionally, the language used to communicate nodes (or elements) between them is based on events. That means, the informational communication between nodes are focused on informational packets that are transmitted throughout the network until the receiver reads the packets and collect/send new pieces of information.

IEEE 1451.X

The IEEE 1451.X is a family of standards for sensor informational communication that was created in *1993* by the IEEE and the National Institute of Standards and Technology (NIST). This standard are categorised on the family of smart transducer interfaces that describes a set of open and network-neutral interfaces for connecting sensors and actuators. This standard communicate networks and processors by using definitions and implementations of trustworthy organisations *(IEEE, 1999)*. Hence, the main aim of this family of standards is to allow access of transducer data through a common set of interfaces whether the transducers are connected to systems or networks via a wired or wireless mechanism.

The main purpose of this standard family is to communicate a variety of sensors with them in order to avoid the wiring between sensors and then, reduce manufacturing costs and efforts in order to orchestrate sensors. Based on this assumption, the main features that the IEE1451 provides are: *(i)* a complete standard that covers a great variety of sensors and actuators; *(ii)* different communication mechanisms that permit to group most of the sensors; *(iii)* compatibility with wired and wireless networks; and *(iv)* a separation of the communication in different layers, simplifying the communication problem.

These characteristics are offered by the definition of a command-and-control architecture that permit to order the sensors and/or actuators to read data, make some specific operation, etc. (see *Figure 2*). General architecture is composed by a "*Network-Capable Application*" (NCAP) that is the element that act as a bridge between the sensor/transducers and the user network. Furthermore, the architecture is composed by "*Transducer Bus Interface*" (TBI) that defines the needed interfaces in order to simplify the sensor/transducer communication. The TBIs that are composed by a transductions, signal processing, conversion tables and data sheets called "*Transducer Electronic Data Sheet*" (TEDS) and a communication module from the IEEE1451.X aligned with the type of communication performed with the NCAP.

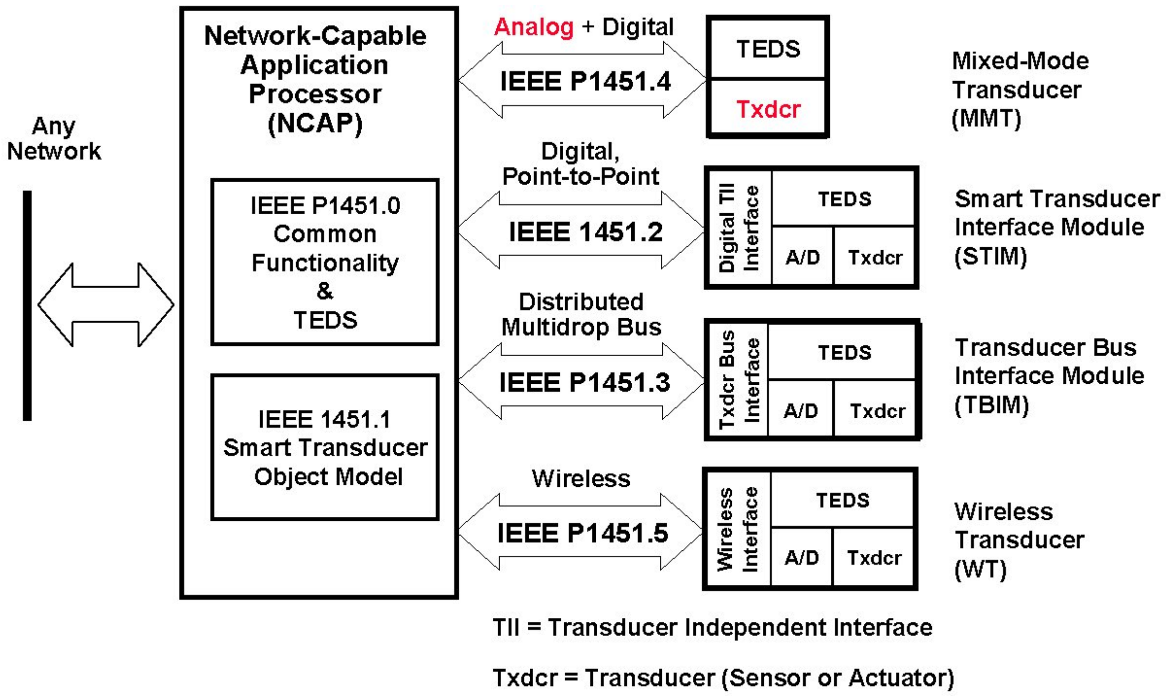

Any Network
Network-Capable Application Processor (NCAP)
IEEE P1451.0 Common Functionality & TEDS
IEEE 1451.1 Smart Transducer Object Model
Analog + Digital
IEEE P1451.4
TEDS
Txdcr
Mixed-Mode Transducer (MMT)
Digital, Point-to-Point
IEEE 1451.2
Digital TII Interface
TEDS
A/D
Txdcr
Smart Transducer Interface Module (STIM)
Distributed Multidrop Bus
IEEE P1451.3
Txdcr Bus Interface
Transducer Bus Interface Module (TBIM)
Wireless
IEEE P1451.5
Wireless Interface
Wireless Transducer (WT)
TII = Transducer Independent Interface
Txdcr = Transducer (Sensor or Actuator)

Figure 2 "IEEE1451.X architecture". Source: (IEEE, 1999)

The communications modules that the IEEE1451 offers to communicate the sensors with the NCAP goes from level *0* to level *7*. The IEEE1451.0 is focused on providing interoperability by the definition of a digital interface that implements the main TEDS commands, functionalities and formats to the rest of the IEEE1451 standards. The IEEE 1451.1 provides a class diagram in order to facilitate the software development and portable for the sensors and transducers. In reference to the IEEE 1451.2, this standards were focused on implementing a common connexion between the TIM and the NCAP in form of "*Transducer Independent Interface*" (TII). The TII is a bus composed by *10* wires that permit to exchange information between the two mentioned entities. This standard was born thanks to the industrial necessity of communicating the transducers in different simultaneous networks.

The IEEE1451.3 has the aim of covering most of the distributed system requirements by the definition of methods that permit to read in a synchronous way a huge variety of transducers. Hence, this standards develop an intelligent transducer interfaces for a "*multi drop*" distributed bus. As contrary, the IEEE1451.4 is focused on defining needed interfaces to communicate analogous transducers. Then, the aim is to incorporate into the 1451 architecture legacy transducers. The IEEE1451.5 is focusing on defining needed communication mechanisms for the wireless sensors by the specification of a "*Wireless Transducer Interface Module*" (WTIM). Regarding with the IEEE1451.6, this standard is concentrating on providing the needed interfaces that permit to communicate sensors via Controller Area Network (CAN)[8] open protocol. Finally, the IEEE1451.7 is aimed at communicating sensors in a Radio Frequency IDentification (RFID) communication environment.

OGC-SWE

The OGC-SWE standards family was born in early *2001* with the aim of integrating and improving the interoperability between sensors located in situ or remotely around the globe *(OGC-SWE, 2011)*. Thus, this family of standards also permits to access sensors information and discover new sensors using the web.

The OGC-SWE standards are organised in several environments (security, environmental information, communication, infrastructure, etc.) covering domain information and linking diverse technologies in these markets with new solutions for plant security, industrial controls, meteorology, geophysical survey, flood monitoring, risk assessment, tracking, environmental monitoring, defence, logistics and many other applications.

The main capabilities offered by this specific set of standards rely on: *(i)* open interfaces for sensor web applications; *(ii)* an interoperability framework for accessing and utilizing sensors and sensor systems in a space-time context via Internet and Web protocols; *(iii)* a set of web-based services used to maintain a registry of available sensors and observation queries; *(iv)* web

[8] CAN protocol definition: http://www.can-cia.org/index.php?id=systemdesign-can-protocol

technology standard for describing sensors' outputs, platforms, locations, and control parameters facilitating it usage across applications; *(v)* encompass specifications for interfaces, protocols, and encodings that enable the use of sensor data and services; *(vi)* quickly discovering of sensors (secure or public) that can meet market needs – location, observables, quality, ability to task-; *(vii)* automatic and standard encoding for obtaining sensor information; *(viii)* readily access sensor observations in a common manner; and *(ix)* subscribe to and receive alerts when a sensor measures a particular phenomenon.

In reference with the general architecture of this family of standards (see *Figure 3*), the OGC-SWE offers a SOA approach mainly composed by a OGC-Sensor Observation Service (OGC-SOS), OGC-Sensor Alert Service (OGC-SAS), OGC-Sensor Planning Service (OGC-SPS), OGC-Web Notification Service (OGC-WNS) and a OGC-Catalog Service. The OGC-Catalog Service is the system in charge of discovering sensors, data providers and services for the catalogued sensors. Hence, this system acts as a dictionary of sensors where specific information about sensors are stored (phenomena, Units of Measure, Sensor Types and applications).

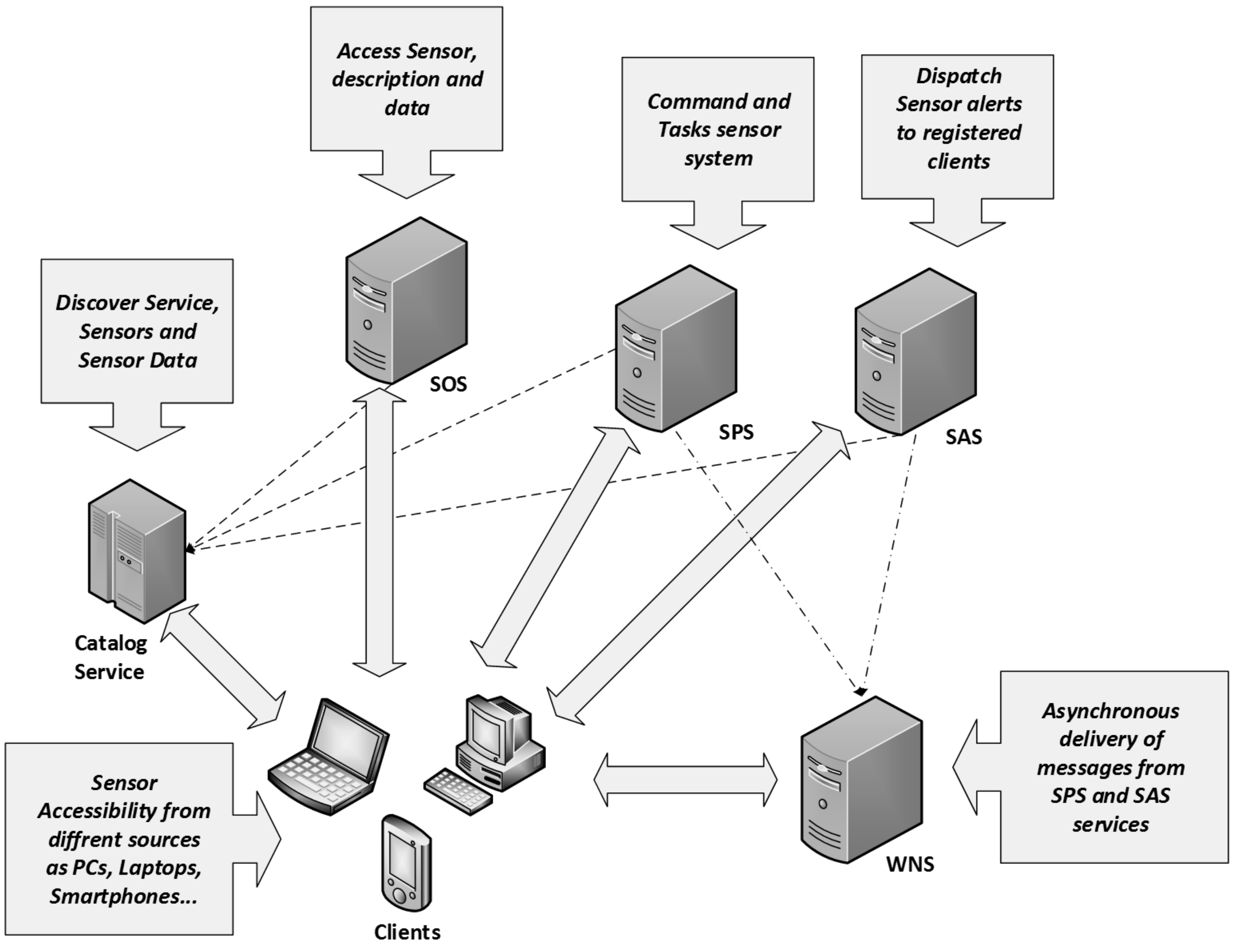

Access Sensor, description and data
Command and Tasks sensor system
Dispatch Sensor alerts to registered clients
Discover Service, Sensors and Sensor Data
SOS
SPS
SAS
Catalog Service
Asynchronous delivery of messages from SPS and SAS services
Sensor Accessibility from diffrent sources as PCs, Laptops, Smartphones...
WNS
Clients

Figure 3 "OGC-SWE architecture"

This OGC-Catalog Service offers needed information from the rest of the architecture about sensor information. Therefore, the OGC-SOS, OGC-SPS and OGC-SAS communicate with this kind of general servers. The OGC-SOS (Data Services) play the role of requesting, filtering and retrieving observations and sensor system information. This element is the intermediary between a client and sensor observation storage technology. The OGC-SPS server is in charge of requesting user-driven acquisitions and observations. This server is the intermediary between a client and a sensor collection management environment. Furthermore, the OGC-SPS permit to schedule operations to gather information from sensors.

The OGC-SAS server (Processing Services) permit to publish and subscribe alerts from sensors. This service permit to manage the alerts relative to sensors of same domain. The OGC-WNS service is aimed at interfacing for asynchronous delivery of messages or alerts from OGC-SAS and OGC-SPS service and other elements of service workflows.

As a transversal part of these main services of the OGC-SWE architecture, the communication languages (or encodings) permit to interoperate with the sensor information by using XML encoded vocabulary. Therefore, XML languages as Geospatial Markup Language (GML) (geo–spatial language), Sensor Markup Language (SensorML) (inter-exchange sensor information), Transducer Markup Language (TransducerML) (language for transducers) and Observation and measurement model (O&M) (categories the observations and measurements), have been defined.

Finally, the OGC-Web Map Services (OGC-WMS) (Portrayal Services) is aimed at publishing geospatial information by using maps as an image in order to produce visual representation of sensors. Furthermore, the continuous evolvement of this architecture has new improvements on other systems inside the presented architecture (see *Figure 3*).

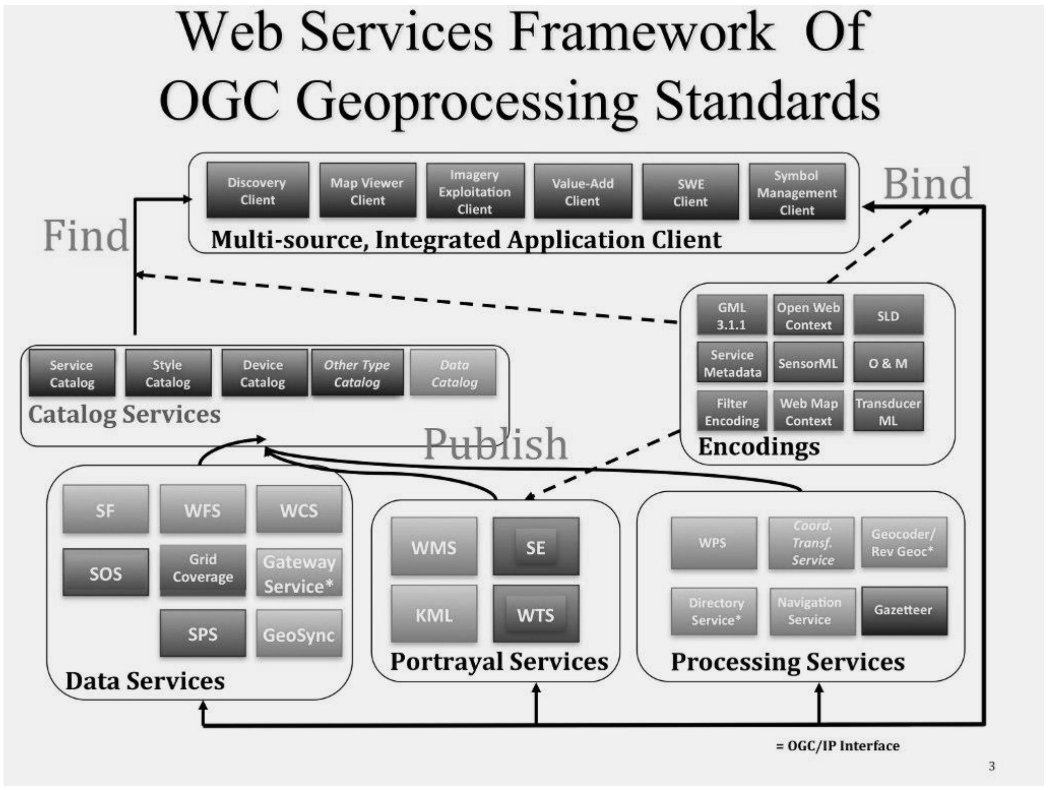


Figure 4 "OGC-SWE Components". Source: http://live.osgeo.org/en/standards/standards.html

Common CBRN Sensor Interface

The Common CBRN Sensor Interface (CCSI) has been defined by a conjunction of organisms such as Department of Defense (DoD), Department of Homeland Security (DHS), Defense Threat Reduction Agency (DTRA), NIST, IEEE and OGC under Joint Program Executive Office for Chemical and Biological Defense (JPEO-CBD). The JPEO-CBD program has the aim of communicating sensors and systems towards the identification of alarms and warnings related with security (e.g. Joint Warning and Reporting Network –JWARN- application)^{9}.

Therefore, this standard is strongly aligned with the security and safety domain by using a Net-Centricity platform (see *Figure 5*). The net-Centricity platform is composed by a

"*Common Modular Communications Interface*" (physical layer) and a "*Net-Centric Security and Discovery Service*" (application layer). The "*Common Modular Communications Interface*" define a standard sensor communications and needed interfaces to discover and include sensors by using a plug-and-play methodology. The identified and compatible sensors are aligned with "*Ground Combat Vehicles*", "*Combat Support Platforms*", "*Naval/Air Systems*" and "*Installation Force Protection*". The "*Net-centric Security and Discovery Service*" provide the needed mechanisms to *(i)* Authenticate service in order to ensure that sensors and applications are authorized to communicate; and *(ii)* Registry services enabling applications to find sensors that will support their operational requirements.

[9] JWARN definition: http://fas.org/man/dod-101/sys/land/wsh2013/204.pdf

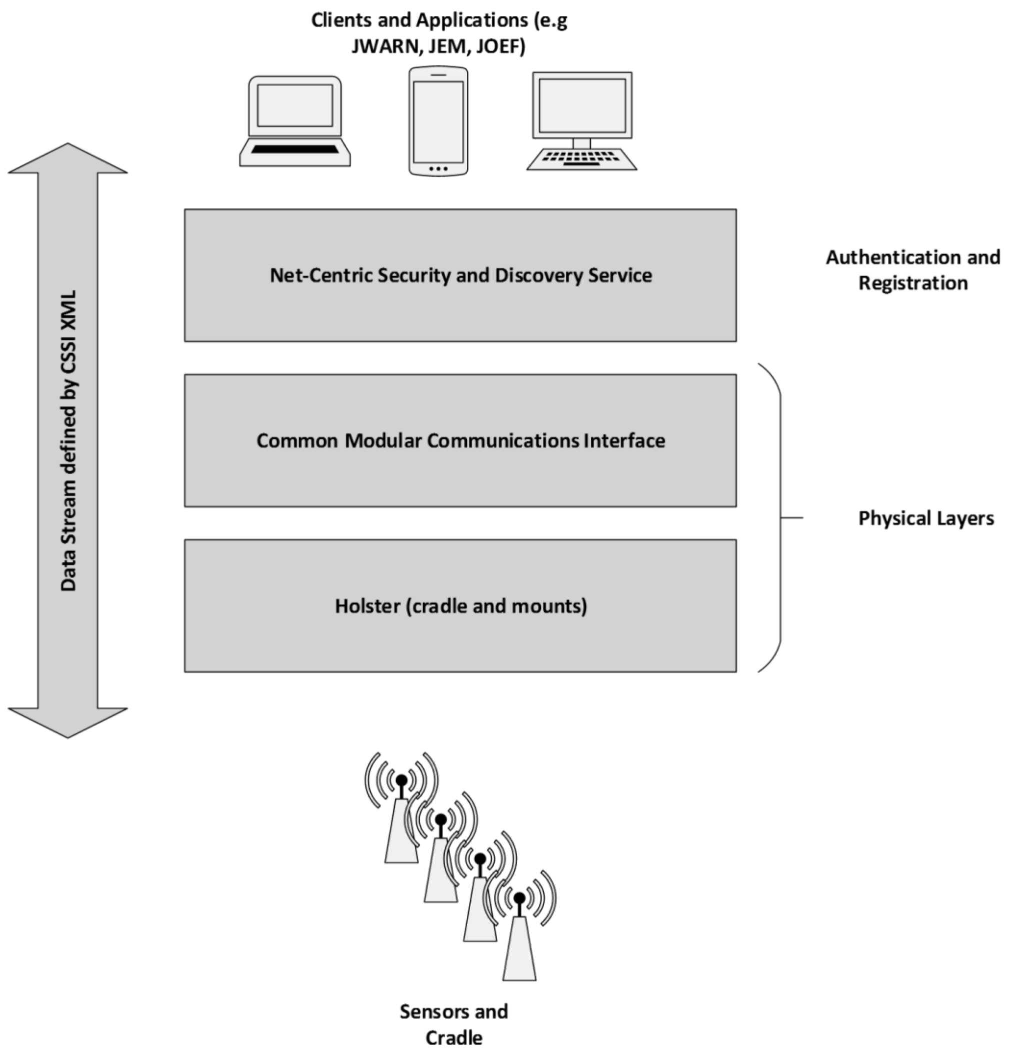

Clients and Applications (e.g JWARN, JEM, JOEF)
Data Stream defined by CSSI XML
Net-Centric Security and Discovery Service
Authentication and Registration
Common Modular Communications Interface
Holster (cradle and mounts)
Physical Layers
Sensors and Cradle

Figure 5 "Common CBRN Sensor Interface architecture". Source: (JPEO-CBD & SSA, 2008)

In reference to the communication between the physical layer and the application layer, the CBRN sensors interface defines an API that permit to integrate both modules into a single system (see *Figure 6*). This API support six types of transactions: *(i)* *Registration* to identify the sensor on the network and handle the sensor's authentication as a valid network participant; *(ii)* *Deregistration* to remove the sensor from the network and deregisters the sensor from any discovery mechanisms; *(iii)* *Connection* to notify the sensor when an application is establishing communication with it; *(iv)* *Disconnection* to break the connection between the sensor and a connected application; *(v)* *Send Message*, to queues a transmission for sending information on the specified CCSI connection; and *(vi)* *Receive Message* to receive a message from a specific connection.

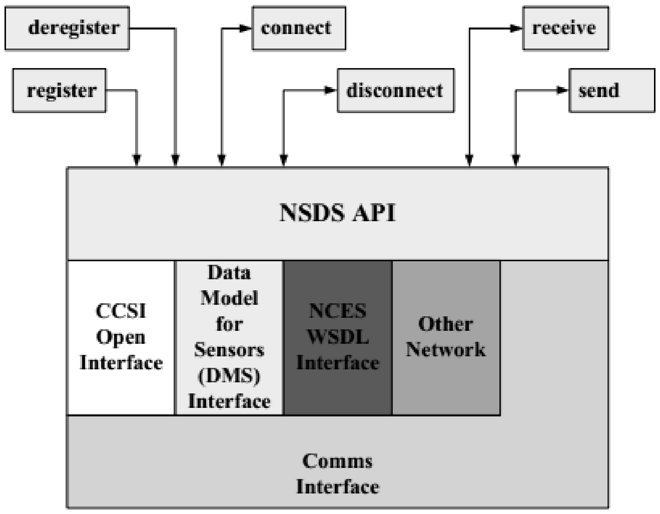


Figure 6 "CBRN sensor API". Source: (JPEO-CBD & SSA, 2008)

The CCSI open interface is a simple Transmission Control Protocol/Internet Protocol (TCP/IP) client-server interface that supports sensor testing. This open interface requires for an initial configuration to be deployed. The Data Model for Sensor (DMS) interface is used within System of Systems Common Operating Environment (SoSCOE) type networks. The Net-Centric Enterprise Services (NCES) Web Services Description Language (WSDL) interface supports operation of the sensor on the Global Information Grid (GIG) or on networks that implement the GIG interfaces.

Additionally, the CBR-CSSI is defined by an XML schema in order to ensure that all required information is provided. Moreover, by the usage of this XML schema also is ensured that all required commands are supported, and that the XML file is syntactically and semantically correct.

DHS-ANSI N42.XX

The N42 was created on *2006* by the NIST. The main aim of this specific standard is to facilitate manufacturer-independent transfer of information from radiation measurement instruments that are used in homeland security applications as well as detection of illicit trafficking of radioactive materials.

The ANSI/IEEE N42.42 standard specifies a XML data format that is used for transferring optional data available by radiation measurement instruments. The schema allows XML parsers to validate the format of instrument data files. That means, the XML schema defines the standard names for data elements and attributes, whether or not they are optional or required for each class of instrument. Furthermore, the XML schema represents the hierarchical relationships between XML nodes for radiation information.

OASIS- Emergency Data Exchange Language

The EDDL *(Raymond et al., 2006)* was created on *2006* by OASIS in conjunction with the DHS. The main purpose of this standard is to facilitate the routing of any properly formatted XML emergency message to recipients.

Then, the XML architecture and design (see *Figure 7*) is mainly formed by a Distribution Element that has been thought of as a "*container*". It provides the information to route "*payload*" message sets (such as Alerts or Resource Messages), by including key routing information such as distribution type, geography, incident, and sender/recipient ids. Hence, the distribution element is formed by a "*Target Area*" that permit to define the geographical position; and the "*Content Object*" that enhance the informational detail about the emergency; and also is able to include into the XML the "*non-XMLContent*" and "*XMLContent*" that serves to encapsulates non-XML contents (e.g. Multipurpose Internet Mail Extensions –MIME- objects) and other XML-Content (e.g. other emergency language).

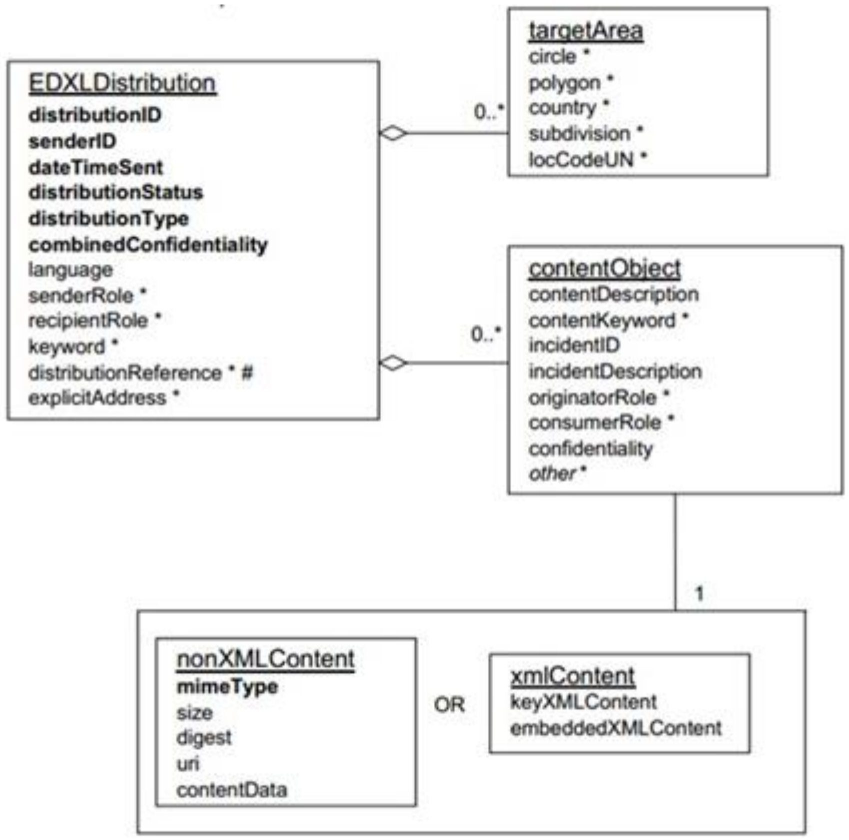


Figure 7 "EDDL XML structure". Source: (Raymond et al., 2006)

Device Kit

The Device Kit is a technology created by IBM on *2007*. The main purpose of the Device Kit is to provide a common interface for the application code to interact with RFID readers and other sensors and actuators.

Based on this main purpose, the Device Kit is focused on providing an OSGI model (see *Figure 8*) that provides support for interfacing with hardware devices. Hence, the Device Kit offer to the developer an abstraction layer in order to include and exchange information with the devices, sensors and actuators. The Device Kit defines an OSGI model based on a *(i)* profile layer, *(ii)* device layer, *(iii)* transport layer, and *(iv)* connection layer.

The "*Connection Layer*" supports to read and write byte streams to the hardware device. The connection does not understand the meaning of the bytes but supports to deliver the output bytes and receive the input bytes.

The "*Transport Layer*" supports to send and receive messages. While the transport layer understands the format of a message, this layer does not understand the meaning of the message. When a device requests a message to send, this part of the standard formats the message into a correct bytes and sends the message. The transport reads input bytes from the connection and parses these bytes into received messages. Furthermore, during the connection, interested devices are notified of the received messages.

In reference to "*Device Layer*", this part of the Device Kit is aimed at interfacing the application and hardware devices. The device layer should shield the application from the low level details of the hardware device. The device layer understands the meaning of the messages and any parameters in a message. When an application executes a command, the command requests the information and sends a command message. In this layer, any signals listeners are notified if any received messages from the transport match the signal messages.

And the "*Profile Layer*" provides to the application a common interface to set of hardware devices. For example, the adapter and profile layer for RFID readers will provide a common interface for the application to set of common functions provided by all RFID readers. This layer uses a publish/subscribe SODA interface. The adapter and profile should shield the application for the knowing which of the common hardware device is being used.

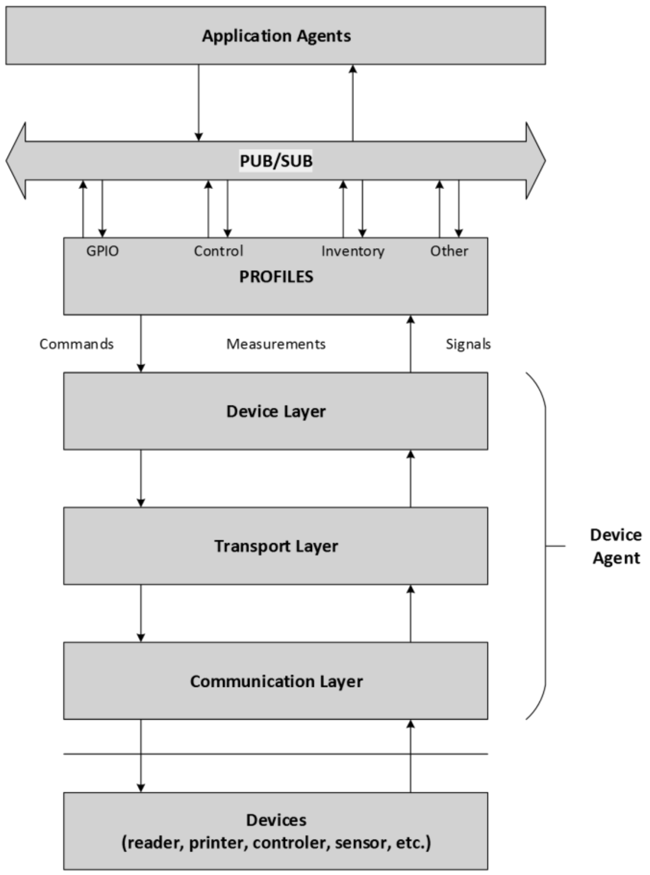


Figure 8 "Device Kit API".

In addition, the Device Kit offer an XML schema called "*Device Kit Markup Language*" (DKXML) that permit to communicate a device agent with a control hardware device by offering the needed mechanism to define device controls, messages and configuration settings. Therefore, this defined XML language has been used transversally for all the Device Kit Layers to transport device information *(Kit, 2013)*.

EDDL

The "*Electronic Device Description Language*" (EDDL) was published by the University of Florida on *1990* (previously called DDL). This communication language was created with the main purpose of integrating technology that uses an electronic file written in a common language. This standard language main motivation is to describe an intelligent device in a machine-readable format to handheld communication with software applications such as a Distributed Control System (DCS) configuration tool or intelligent device management software.

Thus, the EDDL defines needed parameters (data), communication interfaces, user interfaces and operation (Calibration) in order to exchange information and manage sensors devices. This information is exchanged in a plain text that interoperates with an application inside a certain operative system or in a SOA.

Additionally, the EDDL communication language permit to communicate the information with most of the existing systems defined by the FOUNDATION™ technology, HART® Communication Protocol, Profibus, and Object Linking and Embedding (OLE) for Process Control (OPC) Interface.

Discussion over Sensor Languages and Architectures

Most architectures defined for a specific domain rely on using commands and plain text instead of XML. The plain text and commands prevails over XML because the standards rewards transmission and communication speed instead of on data understanding. Furthermore, the command and plain text has achieved more importance than XML in terms of information sharing because of the information is included into a close architecture previously configured and adapted to understand this private way for transmitting information. However, data needs of sharing information between domains and the trends of creating open-architecture and open-data has become the XML communication essential. Hence, information transmitted by XML is the first step to associate meaning to transmitted data and permit to extract the information in a semantically manner.

In spite this standards are being used in different scenarios for different purposes, most cross-domain standard transversal to several domains, is the OGC-SWE. During these last years, the OGC standard has evolved towards empowering the use of the architecture, definition of new interconnected web services for making alerts, collect, perform decisions over the data and creating more XML languages for understanding data in different domains such as Hydrological domain (Water Markup Language), smart city information (City Markup Language –CityML-) or geographical information (GML). However, lacks of semantics understanding on this architecture make difficult to fuse domains information in order to acquire more representative information as alert generation, stakeholders interoperability based on their needs, etc.

Therefore, future trends in sensor languages goes beyond using XML language by empowering data understanding, transmitting sensor information in an open/architecture based on SOA where interested people can include information, sharing information between systems across different domains and making interoperable architecture, facilitating the architecture to be communicated with other systems/devices.

State of the art in semantic sensor web

Nowadays, the main goal of SSW technology is to convert in effective knowledge the great amount of data generated by sensors disseminated everywhere in the world. Mainly, the developed architectures are characterised by *(i)* variability and heterogeneity of data, devices and networks (including unreliable nodes and links, noise, uncertainty, etc.); *(ii)* use of rich data sources (sensors, images, Geographic Information Systems –GIS-, etc.) in different settings (live, streaming, historical, and processed); *(iii)* existence of multiple administrative domains; and *(iv)* need for managing multiple, concurrent, and uncoordinated queries to sensors.

Based on these main architectural features, several authors *(Bizer & Berlin, 2009; Corcho & García-Castro, 2010; Phuoc et al., 2010)* have define the main challenges to be overcame towards achieving a fully and complete SSW technology that permit to create effective knowledge for disaster management (environmental field), homeland security and biotechnological attacks management (security environment), etc. Therefore, the main current research lines in this ambit are *(i)* improving the **abstraction level of the semantic sensors (Challenge 1)**; *(ii)* enhancing the **maintenance, Quality of Services and Security** of the developed architectures **(Challenge 2)**; *(iii)* **Automating data integration, fusion and provenance** during all informational/knowledge process **(Challenge 3)**; *(iv)* **Identifying and locating relevant sensor based data sources (Challenge 4)**; and (v) enabling **Rapid development of applications** that permit to be introduced into an interoperable and open-environment to share knowledge **(Challenge 5)**.

Each of these objectives are separately developed and enhanced by using different technologies. Furthermore, semantic sensor web application should be re-directed towards offering treated knowledge that permit to automatically enhance the generated knowledge and offers to the users more accurate information based on their interests.

Challenge 1: Abstraction Level of Semantic Sensors

The abstraction level of SSW refers to manage, discover, process and gather sensor data from different data sources or environments. Then, the semantic sensor web must to facilitate these tasks by providing syntactic and semantic interoperability *(Arne Bröring et al., 2011)*.

In reference to the **syntactic interoperability**, the efforts has been focused on adopting the OGC-SWE standards and services as main sensor web architecture. This platform defines a series of services and model languages in order to define the process and manage sensors from different environments by using a transversal architecture (see "*State of the art in sensor languages and data exchange*" section). However, substantial effort is required to make a sensor and its observations available on the Sensor Web, since methods and mechanisms to automate this process are missing. In spite of the OGC-SWE define protocols and service interfaces to enable syntactic interoperability, a manual and dependant maintenance is required *(Arne Bröring et al., 2011)*.

Complementing the syntactic interoperability, the **semantic interoperability** permit to define and add annotations into the sensors and sensor data. The semantic interoperability was proposed by *(Sheth & Henson, 2012)* and was focused on combining the OGC-SWE framework with the existing W3C standards for annotating service interfaces and publishing sensor data as RDF (e.g OWL-S usage) or other ontological codifications. After this initial work, other authors has been focused on applying semantic technologies for different tasks related to the discovery and integration of data from autonomous and heterogeneous data sources *(Barbieri, Braga, & Ceri, 2009; J. Calbimonte, Corcho, & Gray, 2010; J.-P. Calbimonte, Jeung, Corcho, & Aberer, 2012; Le-Phuoc, Dao-Tran, & Pham, 2012; Le-Phuoc, Parreira, & Hauswirth, 2012)*. This is the case, for example of the SSN-XG *(Compton et al., 2012)*, OntoSensor *(Russomanno, Kothari, & Thomas, 2005)*, or SemSOS *(CA Henson & Pschorr, 2009)* ontologies to mention a few *(Compton, Henson, & Neuhaus, 2009; W3C, 2011)*. Mainly, these mentioned ontologies are focused on describing sensors and sensor nature towards providing a mechanism to share the information in a standard way. Other research line in this ambit is based on using semantic technology and models to establish a direct connection with the OGC-SWE web services by providing needed syntactic sensor data conversion and server understanding *(Klusch, Fries, & Sycara, 2009; Maué, Schade, & Duchesne, 2009)*. However, semantic descriptions regarding sensor discovery, sensor data retrieval and sensor data publication is still based on a syntactic approach, maintaining the semantic approach as a recommendation inside the OGC-SWE.

As a conclusion of this challenge, semantic enablement is immature for describing and annotating sensor services in the OGC-SWE environment. Moreover, semantics are focused on describing sensor information by avoiding other interesting features as Complex Event Processing (CEP) to analyse stream information from different data sources in order to understand the "*things that happen*", automatic sensor plug and play, discovering and maintenance mechanism of sensors, matching sensor inputs with its correspondent stimuli in order to identify, group and use sensors with similar information, and matching between sensor output an real-world properties towards selecting sensors by expected output in order to make efficient decision in a certain region.

Challenge 2: Maintenance of Quality of Services (QoS) and Security

The maintenance of Quality of a Sensor service and Security is a key aspect in order to provide and share efficient sensor information and knowledge in a certain architecture. This objective relies on assuring data quality and trust during time.

Data quality is a large research area that is not only applicable to sensor-based data, but to any type of data that can be managed in an application *(Esswein et al., 2012)*. Sensor-based data depends largely on the context of the sensor network. In the context of the OGC-SWE, this aspect is taken into account inside model languages such as SensorML. In this vocabulary, data quality measurement is aimed at defining data confidence by a certain process. In other words, data quality measures the level of degradation that a specific sensor measurement suffers during time. Then, this measurement also impact in the measurement and/or calculation of other sensor measurement that uses this information to interpolate different data sources. In spite data quality is taken into account in OGC-SWE model languages, current research lines and enhancements of the standards rely on separating data quality process into a separate model language and structure. Therefore, a transversal data quality measurement for all languages will be established. Tentatively, this definition relies on using the SSN-XG defined by the W3C and the Quantities, Units, Dimensions and Types (QUDT) ontology *(Hodgson, Keller, Hodges, & Spivak, 2014)* in order to be consequent with the defined semantic adoptions and standards. Furthermore, data quality also is supported by the definition of data mining algorithms that permit to avoid data missing by using forecasting algorithms, evade data alterations and outliers, etc. Therefore, the technology used to preserve the data quality is based on defining semantically models and data mining process that facilitates data assurance.

Trust and confidence are becoming key issues in SSW in order to make informed and reliable decisions before acting. Trust and security aspects are transversal aspects needed to share information and connect systems without transferring malicious information in the middle of the transaction. Therefore, current research lines are focused on *(i)* a *local qualitative approach* to referral and functional trust towards using approaches that formalizes reasoning with trust, distinguishing between direct and inferred trust; *(ii) Trust and Trustworthy issues* for Sensor Web by generating models to glean trust information from sensor web to be fundamental enablers for applying semantic web technologies to trust management; and *(iii) Ontology Trust* in order to generate a shared a common vocabulary based on trust definition able of interconnecting and relating aspects referring this concept into a single model and intelligent management methodology.

As a conclusion, data quality and trust inside the semantic sensor web architecture has to be enhanced towards a common management and modelling that permit to assure the data from sensors during it life cycle. This common language or models can be complemented by data mining algorithms to avoid outliers and other data abnormalities. Finally, data trust can be complemented by encryption and hashes algorithms to assure the data has not be manipulated during the transaction.

Challenge 3: Automatic integration, data fusion and data provenance

The automatic integration, data fusion and data provenance refers to the integration of data that comes from different sources and sensor networks, the combination of data with data that persist in external sources in form of static or archived data; and the study and analysis of the observation in order to know the nature of the sensing process.

The integration of data that comes from different sources and sensor networks is focused on understanding the contextual information as for example, corresponding "*Feature of Interest*", spatial and temporal attributes and resources that provides the data and their related services *(Barnaghi & Wang, 2012)*. This aspect is partially aligned with the "*Challenge 1: Abstraction Level of Semantic Sensors*" by the sensor-domain understanding in order to provide a full interoperability between systems inside a wider architecture. This interoperability generates the base to provide an automatic integration of sensors where sensors from different nature can interact in a single environment automatically and autonomously. Regarding this aspect, this challenge relies on generating alignments and semantic metadata between sensor ontologies in order to achieve a situational knowledge.

Another key aspect of this objective is to automatically fuse and process huge amount of continuous data (big data) from sensors from different quality of service, throughputs and geospatial scope. This aspect refers to combine sensor observations and select most suitable information to be presented to the user by taking into account aspects as a quality of service and data, throughputs on data transmission and the location of the sensor. Then, this aspect requires of the development of a situational and or context-aware approach that permit a fully understanding of the environment where sensor actuates. Then, the situational-awareness approach can be achieved by using a semantic and non-semantic approach. In both approaches, main functionalities rely on using techniques for combining sensor data and querying techniques to acquire sensor data.

In the non-semantic approach, techniques such as a data mining, big data and or stream data mining such as Kalman filters, clustering techniques, Bayesian networks and rule-based techniques has been used. These techniques permit to integrate and fuse data from different sources, describe the objects and events, aggregate data, fuse data rules, define thresholds, process data streams in near-real-time at large scale, and measure quality and dynamic issues.

In contrast, the semantic approach relies on using a knowledge-based approach in order to understand sensor behaviour and situational state towards generating the best output or information to a certain situation. In reference to the knowledge base, the usage of mentioned sensor ontologies combined with CEP technology permit to establish a certain process to evaluate a situation of interest *(Hasan & Curry, 2011)*. To enrich the sensor information and situational awareness identification/definition, some approaches uses the Linked Data principle *(Le-Phuoc, Dao-Tran, et al., 2012; Le-Phuoc, Parreira, et al., 2012; Phuoc et al., 2010)*. The Linked Data principle permit to enrich and maintain sensor ontologies and processes with standard information, terms and axioms from other ontologies included in the Linked Open Data Cloud (LODC)[10]. In spite of the possibilities that the ontologies offer, the information/knowledge contained on these ontologies has to be queried and or maintained. In reference to this last aspect, ontologies requires of processing and querying for large volumes of data, semantics descriptions meanwhile efficiency of data storage and data handling is assured *(Jung, 2011)*.

Techniques that are under researching are related with stream querying of semantic data from different data sources (CSV, XML, other ontologies, etc.). In this ambit, techniques such as C-SPARQL *(Barbieri et al., 2009)*, EP-SPARQL *(Anicic & Fodor, 2011)*, SPARQLStream *(J. Calbimonte et al., 2010; J.-P. Calbimonte et al., 2012)* and CQELS *(Le-Phuoc & Dao-Tran, 2011)* are under development and adoption. As remarked in *(J. Calbimonte et al., 2010)*, these techniques are designed for: *(i)* integrating and storing streaming data sources through an ontological unified view; *(ii)* combining data from event-based and acquisition-based streams, and stored data sources; and *(iii)* considering quality-of-service requirements for query optimization and source selection during the integration. Finally, in *(Le-Phuoc, Parreira, et al., 2012)* and *(W3C, 2014)* are presented a comparison between these querying techniques where different streaming scenarios and queries have been benchmarked. In both analysis, C-SPARQL and SPARQLStream are described as the most mature architectures with highly interoperability with Esper and GSN (see *Table 2*).

	Input	**Re-Execution**	**Scheduling**	**Optimization**	**Extras**
Stream SPARQL	*RDF stream and R2RML mapping*	*Periodical*	*External Call*	*Externalised*	*R2RML direct mapping*
C-SPARQL	*RDF Stream &RDF*	*Periodical*	*Logical plan*	*Algebraic & Static*	
EP-SPARQL	*RDF Stream & RDF*	*eager*	*Logic Program*	*Externalised*	*Event operators*
CQELS	*RDF Stream &RDF*	*eager*	*Adaptive physical plans*	*Physical & Adaptive*	*Disk spilling*

Table 2 "Comparison between stream SPARQL processors". Source: (Le-Phuoc, Parreira, et al., 2012) and (W3C, 2014)

As a conclusion, the automatic integration, data fusion and data provenance of sensor information rely on using a semantic and non-semantic approach. Independent of the selected

[10] Linked Open Cloud Diagram: http://lod-cloud.net/

approach, the issues rely on merge sensor information in order to give a proper answer towards a situation based on a specific context.

Challenge 4: Identification and location of relevant sensor-based data sources

The identification and location of relevant sensor-based data sources rely on using sensor information/knowledge depending the zone or the point of interest. Therefore, the main challenge regarding this aspect is the enhancement of the definition of sensor data and it associated observations.

Techniques that explores this idea are focused on using geo-spatial ontologies *(Lieberman, Singh, & Goad, 2007; Lupp, 2008)* towards performing Geospatial-SPARQL (Geo-SPARQL) queries that permit to understand what is happening in a certain zone. Then, geo-spatial ontologies can be merged with CEP *(Sheth & Perry, 2008)* in order to facilitate contextual definition and give a response aligned with the lived reality in this point of interest.

Challenge 5: Rapid development of applications

As a final objective, the rapid development of applications challenge relies on *(i)* making easier the generation of specific application by using same base architecture of sensors; *(ii)* involve the user in order to maintain/evolve the general architecture with a new perspective of sensor information; *(iii)* generate general visualization environments and adaptive tools that permit to be easy extrapolated towards other domains.

Regarding with the generation of specific application by using the same architecture, developed architectures are focused on a Web-based environment. Firstly, a brokered architecture focused on publishing/subscribing communications has been proposed by *(Esswein et al., 2012)* from OpenStack Compute[11]. Mentioned architecture decouple the production and consumption of observation data by using RabbitMQ[12] enterprise bus implementation combined with the widely adopted Advanced Message Queuing Protocol (AMQP). Furthermore, this architecture uses semantic approach over an agent architecture in order to manage the observations that are published. In this architecture, semantics main purpose are: *(i)* ensure well-formed RDF, *(ii)* validate based on OWL/Lite and *(iii)* perform data quality checks. The semantic layer is exploited by OWLMini reasoner cascaded with a second custom rule based reasoner. Furthermore, the used ontologies under the semantic layer are: SSN-XG, Semantic Web for Earth and Environmental Terminology (SWEET) (environmental phenomena)*(Raskin & Pan, 2003)*, W3C-Time *(Hobbs & Pan, 2006)* and OGC-SWE geographic mark-up language (GML-XML Language).

In the architecture proposed by *(Moraru & Mladenić, 2012)* the semantic sensor web is enhanced by *(i)* Analysis of the sensor descriptions and measurements for identifying associated semantic concepts; *(ii)* Extension of the selected ontologies with the specific concepts to the application domain. This mainly implies particularization of the observed properties and features of interest; and *(iii)* Implementation of enrichment components, which are software programs that parse sensor description and measurement. Based on these purposes, the architecture (see *Figure 9*) is based on implementing a Semantic Repository of Sensor Data (SRSD), which contains the enriched sensor descriptions and measurements. Finally, the architecture is also composed by data consumers, such as query end- points, semantic browsers and inference engines.

[11] OpenStack Compute Web Site: http://openstack.org
[12] RabbitMQ Web site: http://www.rabbitmq.com/

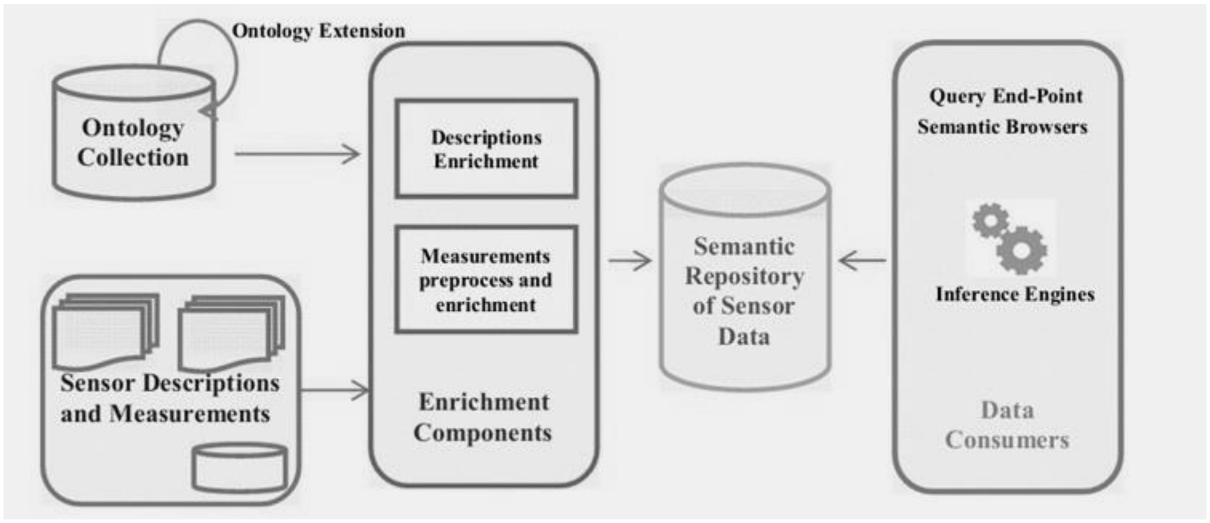

Ontology Extension
Ontology Collection
Sensor Descriptions and Measurements
Descriptions Enrichment
Measurements preprocess and enrichment
Enrichment Components
Semantic Repository of Sensor Data
Query End-Point Semantic Browsers
Inference Engines
Data Consumers

Figure 9 *"*Conceptual framework of (Moraru & Mladenić, 2012) architecture*"*

(Sigüenza et al., 2012) has explored the idea of the MultiModal Architecture (MMI) architecture to manage interesting events based on sensor information. The MMI architecture (see *Figure 10*) considers two important elements: *(i)* a data component which stores data that the Interaction Manager needs to perform its functions; and *(ii)* an event-based communication layer to carry events between the modality components and the Interaction Manager.

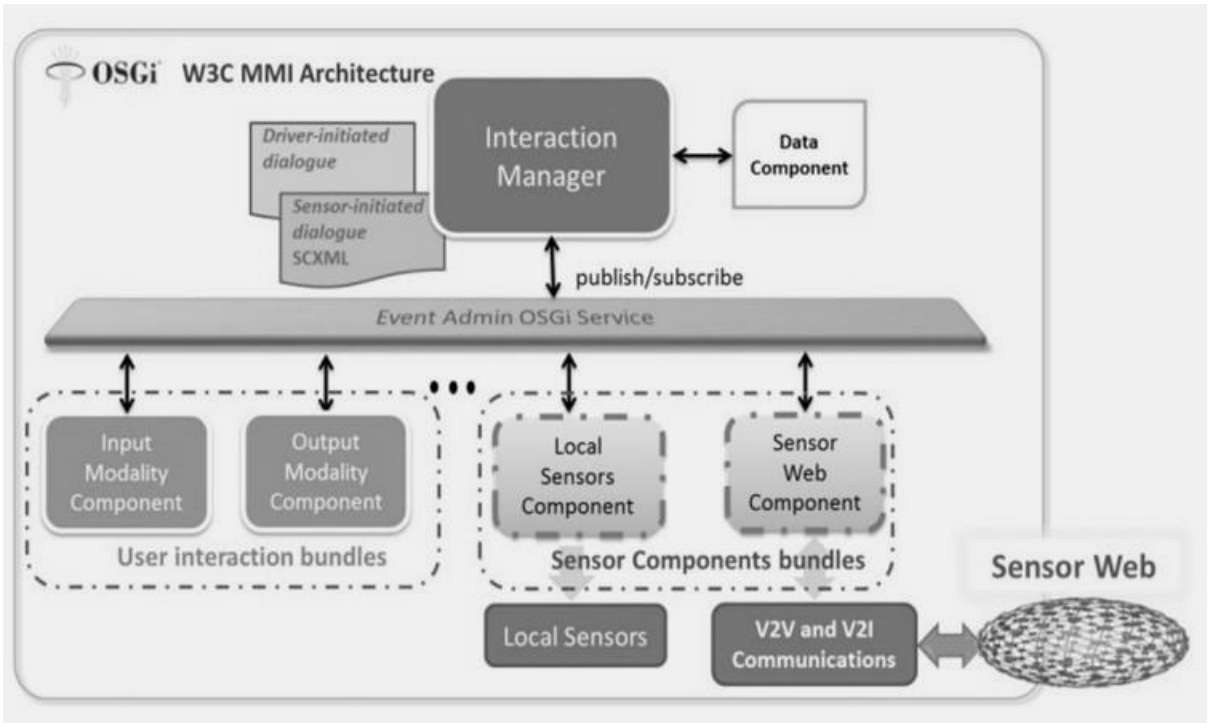
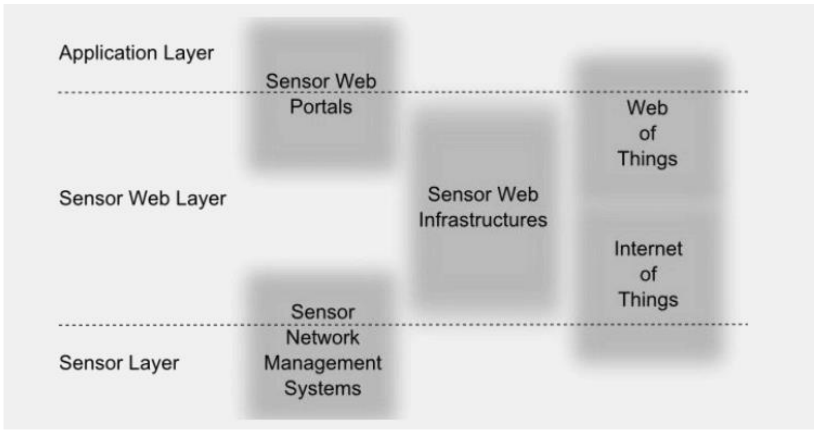


Figure 10 "OSGi MMI Architecture". Source: (Sigüenza et al., 2012)

In contrast to the MMI architecture, *(Broring & Maué, 2012; Arne Bröring et al., 2011)* have researched the idea of creating a semantic sensor web middleware. Then, Sensor Web concept can be considered as a middleware between sensors and applications. Firstly, there is the sensor layer, where actual hardware devices reside and various kinds of proprietary or standardized communication protocols are used by different sensor types (e.g., Wireless Personal Area Network –WPAN- protocols, IEEE *1451*). Intermediary Sensor Web layer provides functionality to sensor resources and applications. The upper part of the architecture is composed by the application layer where direct interaction with clients (human end users or computers) takes place.



Last kind of implemented architectures are grouped into a SOA architecture. First approach depicts the idea of using sensors as a services *(Amato, Casola, Gaglione, & Mazzeo, 2011; Gray et al., 2011)*. Idea of "*sensing as a service*" represents a scalable way to access sensor data through standard service technologies. This approached have received consensus from the community thanks to the deployed of web services to share sensor information. In this architecture, resource is referred to as a device or entity that can provide data or perform actuation (e.g. a sensor or an actuator), and a service is a software entity that exposes the functionality of its corresponding resource. Based on this general assumptions, the architecture (see *Figure 11*) is composed by: *(i)* discovering of data sources and services based on their content and spatio-temporal coverage; *(ii)* accessing and manipulating sensor and related data in near real-time; *(iii)* on-the-fly integration of multiple heterogeneous sensor and stored data sources; and *(iv)* multiple conceptualizations of data.

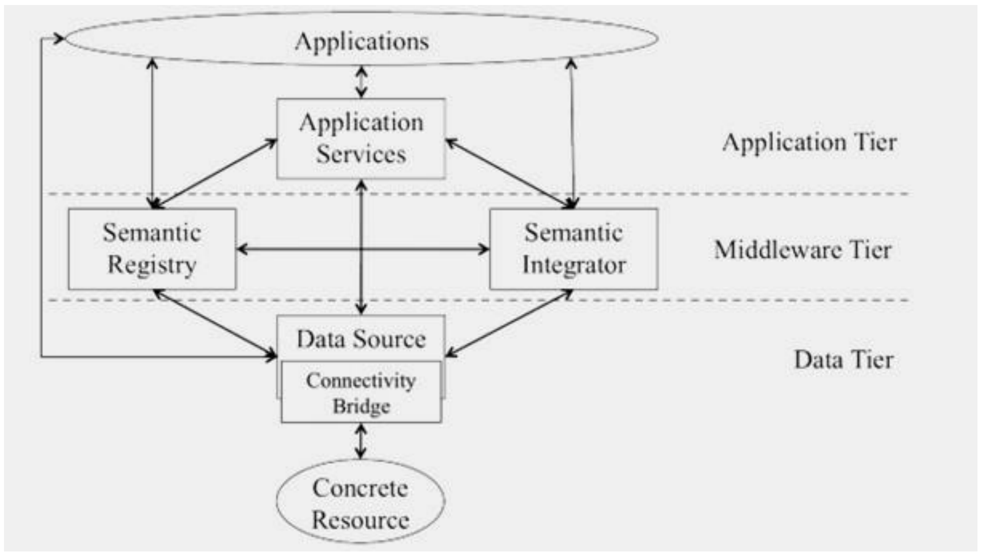


Figure 11 "Sensor as a Service architecture". Source: (Gray et al., 2011)

Second approach is focused on a virtualization framework that is capable of getting and making available information from different sources and services in from of virtual services *(Alam, Chowdhury, & Noll, 2010)*. Moreover, this architecture provides a web service interface for functional aspects of IoT cloud's connected objects. The architecture is composed by three layers (see *Figure 12*): *(i)* the Real-world access layer, *(ii)* the Semantic Overlay layer, and *(iii)* the Service Virtualization layer. The Real-world access layer provides an interface to underlie IoT cloud. One of the main goals of this layer is to get real-world information and carrying this information to the upper layer for further processing. This layer can also transfers action messages from upper layer and then selects appropriate adapters to deliver it to the IoT cloud's connected objects and actuators. The semantic overlay provides the semantic model of underlying IoT cloud by maintaining an IoT ontology, the sensor ontology, the event ontology and the service access policies. This layer is also capable of importing any sensor system description in SensorML and translates it to OWL description by using the sensor ontology and the mediation rules. The goal of the service virtualization layer is to expose functional aspects of underlying IoT cloud and information in the form of services. This layer aims at delivering requested information from the user based on their access rights.

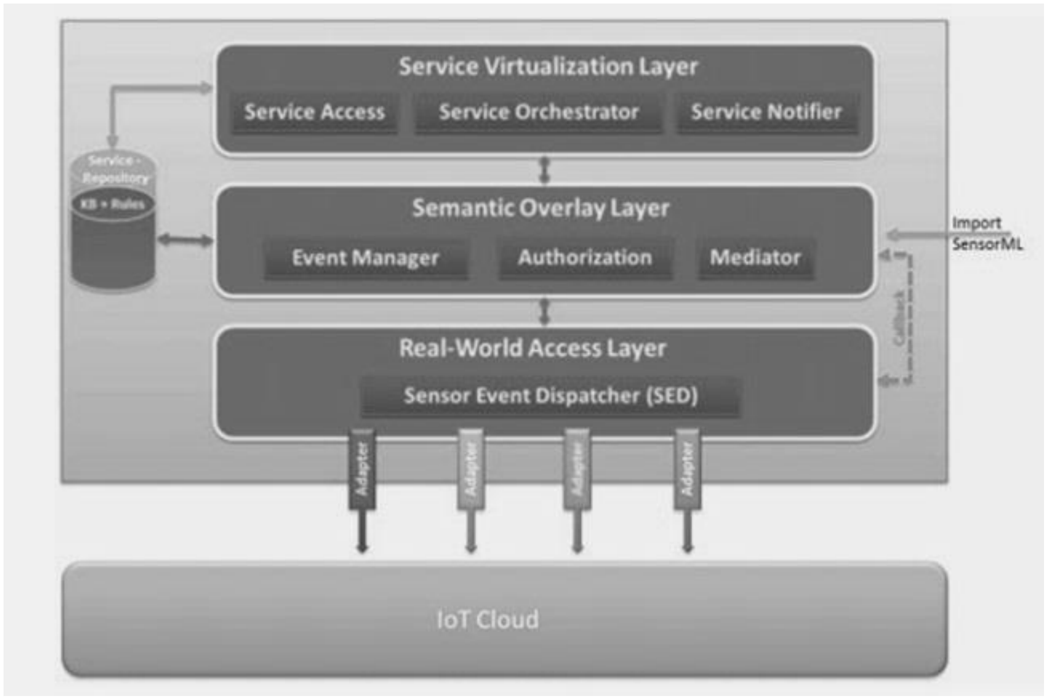


Figure 12 "IoT Sensor Cloud Architecture". Source:(Alam et al., 2010)

In spite these depicted architectures are general and can be implemented in various domains, a common issue of presented architectures relies on the avoidance of user involvement as a part of the systems. That means, in the mentioned architectures, the user only is capable of asking and receiving information referring to the sensors or, in the most complex architecture, alerts and recommendations as a response of a detected event. However, the user can offer much more knowledge to the system as, for example, introducing information that can be annotated into the semantic layer of the architecture, validating the information, annotating sensor results in order to identify other relevant events, etc. In reference to this aspect, latest architectures introduce the concept Humans as a Sensors (HaaS) or the Volunteered Grafical Inforamtion (VGI) as a new sensor *(Sigüenza et al., 2012)*. Therefore, by using mobile applications, the user is capable of introducing into the system relevant information that can be added and proposed to other users.

Additionally, other important aspect of the architecture is the visualization engine for sensor information. In this ambit, the visualization environments are motivated on creating adaptive and intelligent tools capable of being extrapolated to other domains. Aligned with this aspect, current research lines are focused on visualizing the information graphically through EXHIBIT *(Exhibit, 2012)* or RIZHOMER *(García & Brunetti, 2011; García, Gimeno, & Perdrix, 2008)* tools. This kind of tools permit to load an ontology or structured data and visualize it in different formats (e.g. timeline, map, etc.) depending of the kind of information to be showed. Furthermore, this kind of visualization tools represent the information following the HTML 5 basis *(Berjon et al., 2014)*. In reference to HTML visualization of the ontology, standard languages as an RDFa and JSON-LD are being established as a structured languages for transferring information between Webs and applications. In case of RDFa, this lightweight language permit to deal with linked data by using an XML format by following the RDF notation (subject, object and predicate). Furthermore, this language can be included into XHTML or HTML and facilitates search engines to link and understand the information. Likewise, this kind of language is highly recommended to transfer semantic information in a SOA architecture because of it is highly XML based formatted.

Referring JSON-LD format, this language is a natural extension of the JSON format and permit JSON data to be interoperable at web-scale. If RDFa is associated to SOA architecture, JSON-LD usage ambit is focused on REST web services and unstructured databases.

As a conclusion of this part, visualization and generation of adaptive tools is a research line that needs reinforce in order to include the user in the sensing loop. Most of the authors have been researching how to intelligently show the information into a web based on user requirements *(Brunetti, García, & Auer, 2013)*. However, user can play an important role in semantic web application because it feedback can permit to discovery new knowledge, automatically maintain the system and define new rules or relations between concepts that facilitates the enhancement of the information of the domain or other similar ones (holistic approach).

Discussion over the Semantic Sensor Web current situation

During this section has been presented the main challenges to be overcame during the development of a SSW architecture. Based on this identified challenges, a general architecture based on web services (SOA or REST) and the Semantic Sensor Web concept can be extracted (see *Figure 13*). Mainly, this architecture is composed by *(i)* Data layer; *(ii)* Semantic Layer; *(iii)* Query Layer; and *(iv)* Visualization Layer.

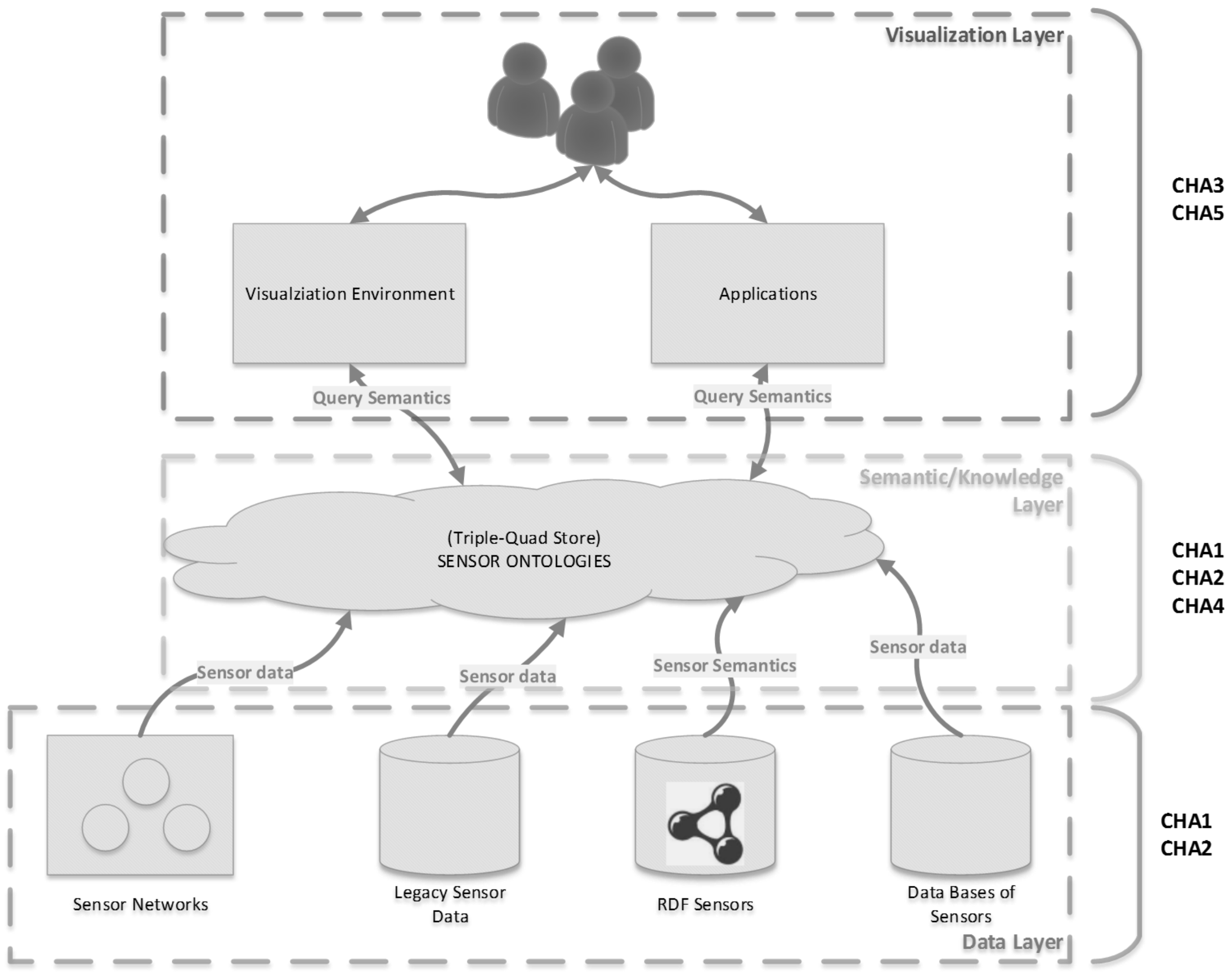


Figure 13 "General architecture extracted from the challenges"

Data Layer is aimed at registering sensors in the architecture from different data sources as, for example, sensor information, legacy sensor data, other ontological resources (RDF) and data bases of sensors. Therefore, this layer corresponds with the **Challenge 1** and **Challenge 2**. So, this layer includes as a challenges the development of sensor networks by using a standard communication language (OGC-SWE standards), the generation of specific sensor description for the sensor network and the definition of data quality models by also including data mining algorithms to remove outliers from observations and other relevant patterns that come from Stream data mining o big data analysis.

Once the sensor information has been registered in the architecture, the **Semantic Layer** is focused on managing observation and measurement procedure of sensors including also other ontologies or terms of the Linked Data. Then, sensors can be understood and a higher treatable information that can be given to the user. The technology aligned with this part of the architecture is focused on overcoming the **Challenge 1**, **Challenge 2** and **Challenge 4**. Therefore, the main objective are to construct ontologies that includes sensor description, quality models, time and measurements ontologies, complex processing ontologies, geo-SPARQL and Stream SPARQL compatibility.

The defined semantics are queried and reasoned by a SPARQL end-point in the **Query layer**. Then, the sensor ontology has to provide SPARQL end-point for accessing to the semantic information. In this ambit, the SPARQL end-point may depend of the informational source, enabling the usage of technology such as Data to Relational Query (D2RQ)/ Relational Data Base (RDB) to RDF Mapping Language (R2RML) for semantic accessibility to data base information; Streaming SPARQL end-points to semantically provide information based on continuous streams of data (from web-sites, CSV files, etc.); and Triple (Quad)-store SPARQL end-points in case of semantic information would be saved in a semantic repository as SESAME, VIRTUOSO or similar approaches *(Boncz & Pham, 2013; Cudré-Mauroux & Enchev, 2013)*. Generally, this part of the architecture has to give to the user the possibility of interacting with the ontology by *(i)* querying the ontology knowledge, *(ii)* annotating to the terms needed extra information and *(iii)* generate new knowledge that comes from the interaction with the visualization environment. Hence, this layers corresponds with the challenges defined in under **Challenge 3**.

The higher layer of the ontology corresponds with **Visualization layer**. This layer is aimed at interacting with the user by intelligently showing the information from the sensors, tagging the information from the sensors in order to obtain feedback from the user, and creating more ontological annotation over the sensor results in order to reinforce the knowledge generated by the ontology and the data mining process (or big data or stream data mining). Hence, this part of the semantic sensor architecture is aligned with the **Challenge 5**.

As a conclusion, the presented general architecture creates a roadmap for the semantic architectures based on the SSW concept. Furthermore, this architecture also identifies that most of SSW work has been focused on the lower part ("*Data Layer*" and part of the "*Semantic Layer*"). However, more efforts have to be applied into the "*Semantic Layer*" to deal with a big-data environment. Furthermore, efforts have to be focused on the "*Visualization Layer*" that will permit to involve the user into the process of knowledge/wisdom generation.

OBJECTIVES

Based on the main challenges identified on "*State of the art in semantic sensor web*" section, the present part of the document is pointed at depicting the main objectives that the exposed research is going to overcome. Mainly, the general objective for the current research is to design an **architecture that provides a model, aligned with current standards, to manage sensor web information providing a knowledge layer capable of discovering rules, make recommendations and use user feedback to learn about the information captured with the aim of generating new knowledge and transforming it into wisdom *(Corchero et al., 2013)*.**

In practice, this general objective has been subdivided into different small objectives that permit to modularise the architecture and facilitates the continuous development of mentioned architecture. Hence, specific objectives of the designed architecture are:

- **Objective 1 (OBJ1) - "*Development of a Sensor Network*"** based on standard information that permit to communicate information with most of the sensors. Then, this sensor network has to be based on the OGC-SWE standards that are emerging as a sensor communication language (XML-based files for transporting sensor information).
- **Objective 2 (OBJ2) – "*Develop a core semantic layer*"** by using a set of ontologies that assure sensor description, OGC-SWE compatibility, geo-spatial reasoning, representation of sensor data quality as well as assure the annotation of sensor from user feedback and understand sensor information patterns that comes from a stream reasoning as well as big data analysis.
- **Objective 3 (OBJ3)- "*Development of data mining or big data layer*"** that permits to remove noisy data from the captured sensor information as same time as patterns are identified over raw sensor information. This identified patterns can serve to the ontology

and the general architecture to generate proper information, alerts, recommendations and actions to the user.

- **Objective 4 (OBJ4) - "*Development of a visualization environment*"** that facilitates the sensor data representation and configuration meanwhile the user enhances the discovery knowledge by annotating information over data and then, producing wisdom (or robust knowledge).

DESIGN OF THE ARCHITECTURE

Derived from the proposed objectives, the designed architecture (see *Figure 14*) has been defined by following the main challenges and the current trends which semantic sensor web concept is focused on. Largely, the proposed architecture is based on a web service architecture based on REST web services in which legacy data bases, sensor ontologies, data bases of sensors as well as OGC-SWE sensor information and observations, are fused into a triple-quad store. This triple-quad store layer permit to access sensor information and it observation by using two end-points: *(i)* a SPARQL end-point to query sensor information with low time variability (e.g. senor descriptions, measured phenomena, etc.) and reason over the region that a specific sensor is located; *(ii)* a SPARQL streaming end-point in order to query and reason over observed sensor information in form of streams or "*events*" on the fly, giving to the architecture the dynamicity and the capacity to reason and act in an on-line mode. Complementing semantic information access and sharing, a visualization environment that permit to deal with the user, is a highlight point of the architecture. In the proposed architecture, user interaction will be done by generating semantic annotation over collected data in order to generate, remove and upload rules that permit to identify critical events over data.

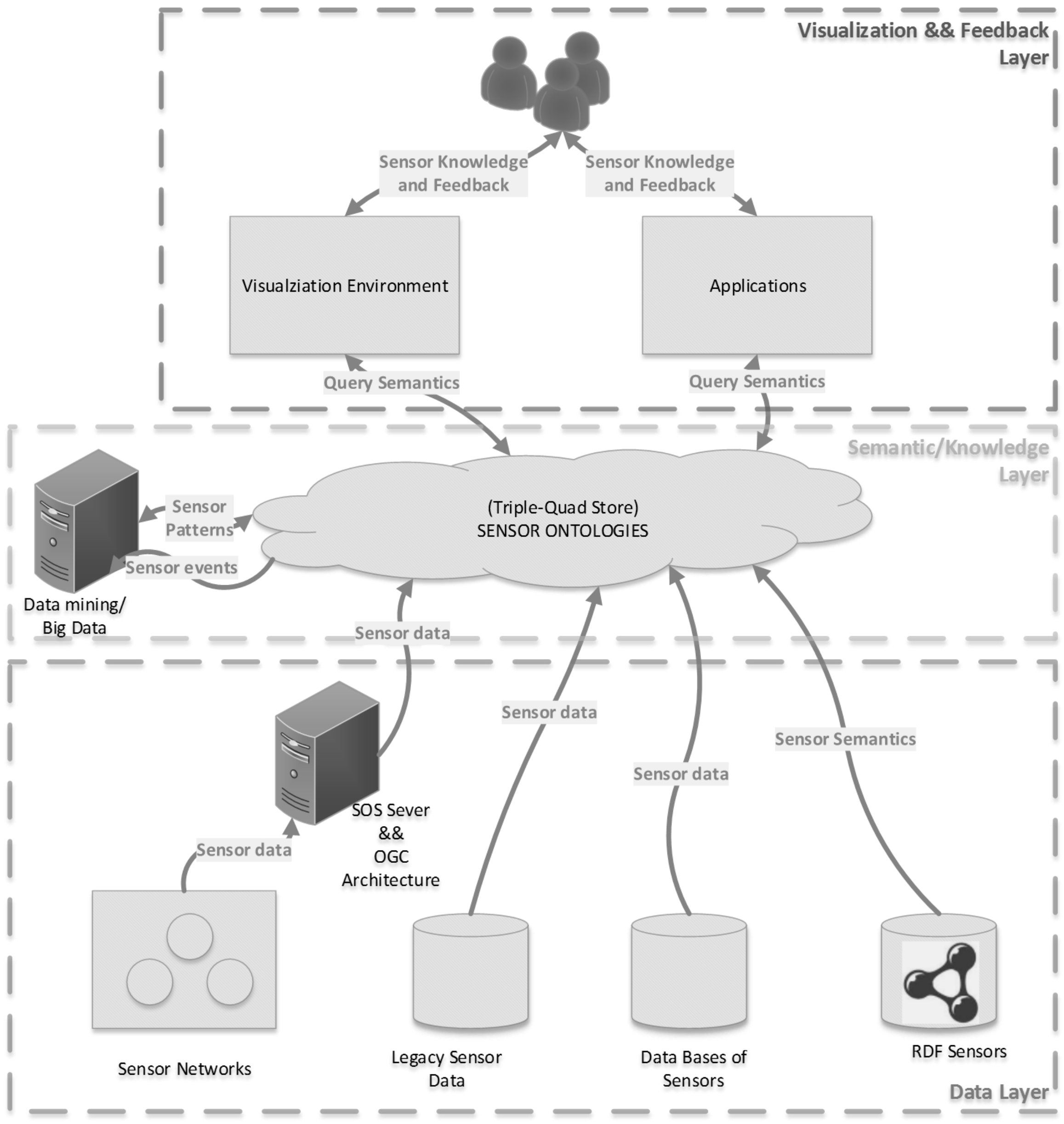


Figure 14 "Designed Semantic Sensor Web architecture"

Detailing the architecture, the proposed design is an extension of the architecture depicted in the "*Discussion over the Semantic Sensor Web current situation*" section. Therefore, the designed architecture is composed by the following layers:

(i) A **Data layer** that is focused on gathering information from a set of sensor networks by following the OGC-SWE standards, databases and/or sensor-web sites. Furthermore and taking into account the importance of the users, the architecture is able of gathering information from the VGI or Humans-as-a-sensors (Twitter, YouTube, Facebook, etc.). Then, sensor information from the environment can directly be fused with relevant information that comes from "*events*" or streaming information from users connected to Internet.

(ii) A **Semantic Layer or Knowledge Layer** is aimed at creating knowledge approach based on the sensor information, it observations and other relevant information from Internet. The knowledge in the architecture is represented and managed throughout knowledge bases. By supporting the defined knowledge-bases, semantic annotations over the stored knowledge will be generated by two approaches. Firstly, by the application of a data mining, stream data mining and Big Data analysis over the raw sensor information collected from the source. Based on the extracted knowledge from raw data analysis, RIF or SPIN rules will be created in order to include data analysis result into the architecture. Secondly, the knowledge also can be generated by performing reasoning over the streaming information during the execution of streaming queries. One of the main advantages of the architecture is motivated by the storage of relevant sensor information (patterns) that permit to identify and understand anomalous or crucial events (patterns about critical events).

(iii) A **Visualization and Feedback Layer** that interacts with the user to give them information about sensors, alerts and recommendations. At same time this layer is visualising sensor-related information, the visualization engine is able to get feedback from the user in order to reinforce the discovery knowledge from semantics layer. As the architecture is based on REST web services, all needed semantic information for the visualization, as well as the annotation performed by the user, will be transferred through the architecture using JSON-LD standard language. Then, by using this language to transfer sensor information with Internet and the user, external applications can also interact with the presented approach by the corresponding standard JSON adapter developed in most of programming languages. Thus, the rapid development of applications is partially accomplished by using a standard languages that can be understood by applications without spending too much additionally efforts.

EARLY DEVELOPMENT OF THE ARCHITECTURE

By taking into account the designed architecture (see "*DESIGN OF THE ARCHITECTURE*" section), the main aim of the present section is to depict the development done during this last months in the way of constructing a fully semantic sensor web architecture. As the coverage of the designed architecture is bigger than the current Master Thesis, the developed architecture covers the **Objective 1** and **Objective 2** of the proposed semantic sensor web objectives by (see "*OBJECTIVES*" section):

- The **definition of a semantic structure** capable of representing sensor information, observations that sensor perceives as well as the time series that the sensor collects. Complementing sensor information and it observations, a semantic layer is capable of representing events, people/organizations in charge of the region where sensor actuates, etc. For that reason, inside the semantic layer has been included concepts related with this kind of information. Additionally, the semantic layer has been expanded by a highly compatibility with RIF and SPIN rules by including needed vocabularies. Moreover, a highly compatibility with OGC-SWE language has been reached by defining semantic resources related with information stored in the OGC-SOS servers.
- The **constructions of adapters** that permit to gather information from different data sources. The adapters corresponds with the usage of D2RQ/R2RML technology in order to abstract a data base into a semantic layer an then, access to the information using SPARQL end-points without replicating the information in various servers or semantic triple-stores. Moreover, an Esper adapter has been built in order to include into the architecture information that comes from data streams such as CSV, websites or databases.

- The **incorporation and adaptation of streaming reasoning and querying engine** that permit to select only relevant information that comes from the result of fusing different data sources in an online mode. In order to accomplish this aspect, SPARQLstream engine (currently called morph-streams) and C-SPARQL engine has been taken into account.

With reference to the development aspects of the architecture, all the adapters and code have been implemented in a Java and a Gradle module has been used to manage dependencies and the building process. Furthermore, Gradle allows defining tasks for code quality improvement and facilitates code development and maintenance. Additionally, the code has been making accessible in a GIT repository[13] in order to accept contributions once the development of the architecture achieve a more mature state.

Regarding with the ontology development, the framework selected relies on Top Quadrant software called TopBraid-Composer in it free version[14]. This software has been selected against Protégé thanks to the Eclipse interface and the highly compatibility with SPIN rules and reasoning that it provides.

Complementing this development environment, a virtual machine that includes Geo-Spatial tools and services has been used for testing and sensor data integration. This virtual machine is called "*OsGeo Live*"[15] and it includes an installation and deployment of OGC-SOS server based "52º North" approach. Additionally, the virtual machine also includes some initial

[13] GIT repository of the development: https://github.com/aolite/semanticsensorProject

[14] Top Braid Composer Web Site: http://www.topquadrant.com/tools/ide-topbraid-composer-maestro-edition/

[15] OSGEO Live Virtual Machie: http://live.osgeo.org/es/quickstart/virtualization_quickstart.html

data sets that has been used for testing the developed application. In order to make the tools and services accessible, the virtual machine has been installed into a server that be accessed by the "*http://osgeo.gerardfont.cat/*" URL.

Semantic Structure Definition

The semantic structure defined (see *Figure 15*) has been mainly based on the **SSN-XG** ontology. This ontology has been selected against others because of the high description of sensors and the partially alignment with the OGC-SWE sensor web approach *(Compton et al., 2009)*. The SSN-XG ontology mainly represents the "*Observation*" that sensor collects (pair of date time and value), the "*FeatureOfInterest*" or entity that measures the observation (physical or abstract), and the "*Property*" (or phenomena) that the observation represents. In spite SSN-XG ontology offer a complete representation of the sensor domain, this ontology diverges from the representation of the entities or domain proposed by the OGC-SWE in the OGC-SOS sensor in the "*Observation and Measure*" concept that is more detailed and realistic in the OGC-SWE models.

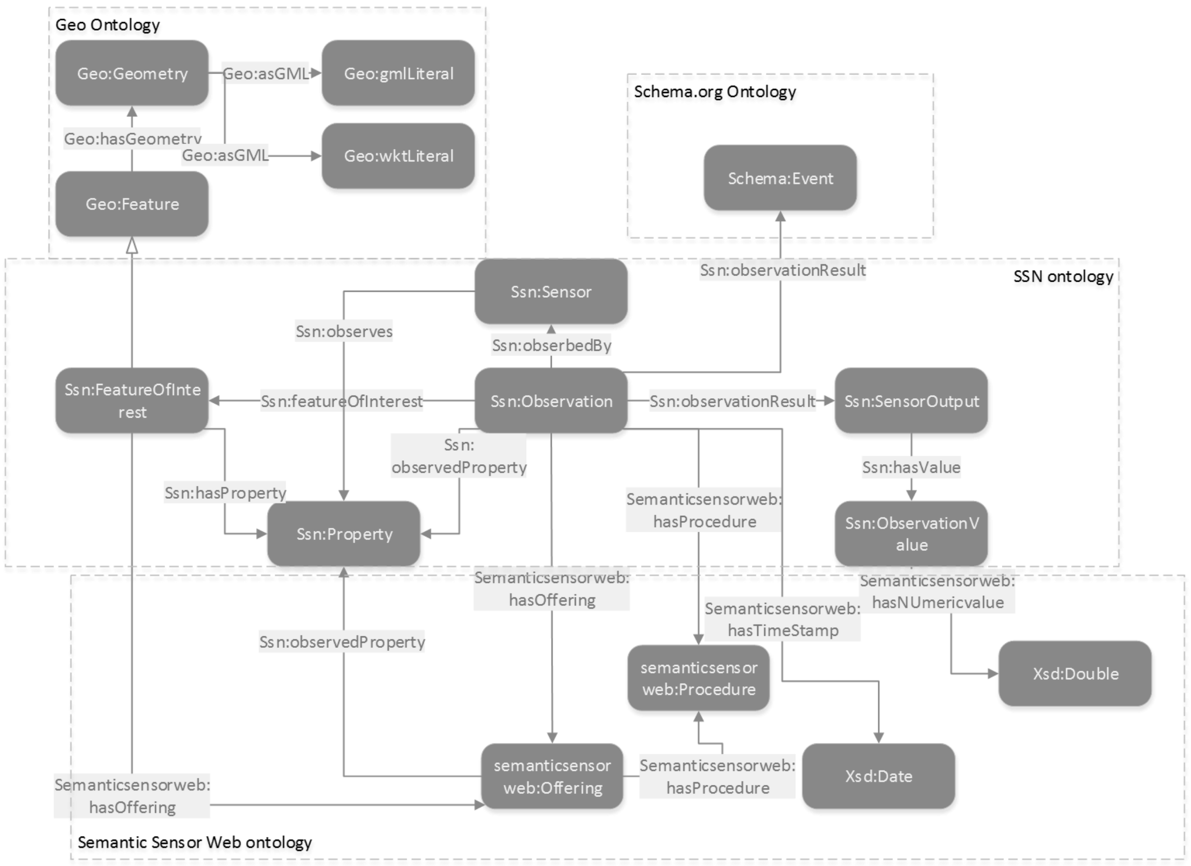


Figure 15 "Schema of the designed ontology"

In order to align the SSN-XG ontology with the OGC-SOS architecture, several standard ontologies as Geo-SPARQL schema ("*Geo ontology*") and "*schema.org*" ontology, have been included. In the one hand, the Geo-SPARQL ontology permits to perform geo-spatial reasoning over the axioms defined in the ontology. The only requirement to perform this kind of reasoning is to make as a child of "*gml:Feature*" all entities that are associated with a geo-spatial reference coded in GML or Well Known Text (WKT) notations. On the other hand, the "*Schema.org*" ontology permits to represent events in the ontology that comes from external sources as a Social Networks. Hence, by the incorporation of this ontology, user annotations can be represented in form of events.

Additionally to mentioned ontologies, specific ontological resources have also been included. This concepts are under the "*semanticsensorweb*" URI[16] and represents those entities, properties and data properties that are represented in the OGC-SWE domain model and not covered by any of mentioned standard ontology. This is the case of the entity "*Offering*" that permits to aggregate as an observation (abstract observation) a composition of an "*Observation*", "*FeatureOfInterest*", "*Phenomena*" and "*Procedure*". Another example is the case of the entity "*Procedure*" that represents the process followed to generate the observation (e.g. "*direct measurement*", "*estimation*", "*forecasting*", etc.). In reference to the included object properties, needed properties regarding with "*Procedure*" and "*Offering*" has been built to link these entities with the rest of domain model (e.g "*hasProcedure*" and "*hasOffering*" relationships). At last, data properties has been also defined in order to associate to the defined semantic entities: a name ("*hasName*"), an id ("*hasID*"), and a description ("*hasDescription*"). Specifically for certain entities, "*hasValue*", "*hasDate*" and "*hasUnit*" data properties to represent the values/dates/units have been associated with an "*Observation*" entity; "*hasSchema*" and "*hasLink*" properties to define an URL to annotate into the model a XML schema, has been also included; and the "*hasType*" data property that permits to associate an entity with the type of value that it represents (e.g. MIME, URL, XML, etc.). Additionally, inverse properties have been defined when required in order to give more richness to the ontology and facilitate the querying and reasoning.

As a conclusion, the designed ontology represents in a high level all the domain model that are covered in the OGC-SWE sensor web approach. This has been achieved by using

[16] Semantic sensor web URI: http://www.udl.griho.cat/semanticsensorweb

standard ontologies from the W3C and specific concepts to complement the gaps and unions between the used standard domain knowledge-bases.

Incorporation of streaming reasoning and querying

By taking into advantage the defined semantic domain, next development stage has been focused on establishing needed mechanism to access and reason over sensor information by using static SPARQL and stream SPARQL queries. Based on the existent streaming reasoning and querying engines (W3C, 2014), only "*SPARQLStream*" (currently called "*morph-streams*") and "*C-SPARQL*" stream are capable of performing SPARQL and Stream SPARQL in the same engine as well as they are the only engines able to be extended towards REST web-service are. These both capabilities are essential requirements for our approach and it extensibility.

The initial intention was to integrate into the architecture the "*SPARQLStream*" or "*morph-streams*". This stream engine was developed by the Polytechnic University of Madrid (UPM) on *2012*. The main strength of this streaming architecture is the capability of merging information that comes from GSN virtual sensor library and Esper library to model CEP over collected data. Another strength of the architecture is the usage of R2RML mappings in order to include streams from data bases of sensors. Furthermore, the architecture offers the possibility of extrapolating the console streaming engine towards a web-based by using Play web service based on Scala framework. However, the development of this architecture was stopped one year ago and the previous and latest stable developments have the weakness of rigidity to the examples that the UPM provides. Moreover, stable versions of "*morph-streams*" also generate incompatibilities between runtime and development libraries, and also incompatibilities between the Scala framework in which morph is developed and JAVA framework (our based approach).

Furthermore, the community and documentation of the library is scarce, aspect that difficulties the integration of the approach with external architectures.

Therefore, the development has been derived through the usage of **C-SPARQL**. This streaming engine is fully based on a JAVA framework where the main based-libraries are **Jena** for semantic manipulation, and **Esper** for event integration. Another key aspect of this architecture is the development and easy extensibility to the usage of a REST web service. This extension is provided with the C-SPARQL engine. Regarding with C-SPARQL documentation as well as it derived libraries are in abundance thanks to the high community support and the fully demos provided by the authors. However, the main inconvenient of this architecture is that the adapters to deal with databases and sensor web pages has to be defined and registering into the platform specifically. That means, in the community there isn't exist any general tool that actuates as a C-SPARQL adapter.

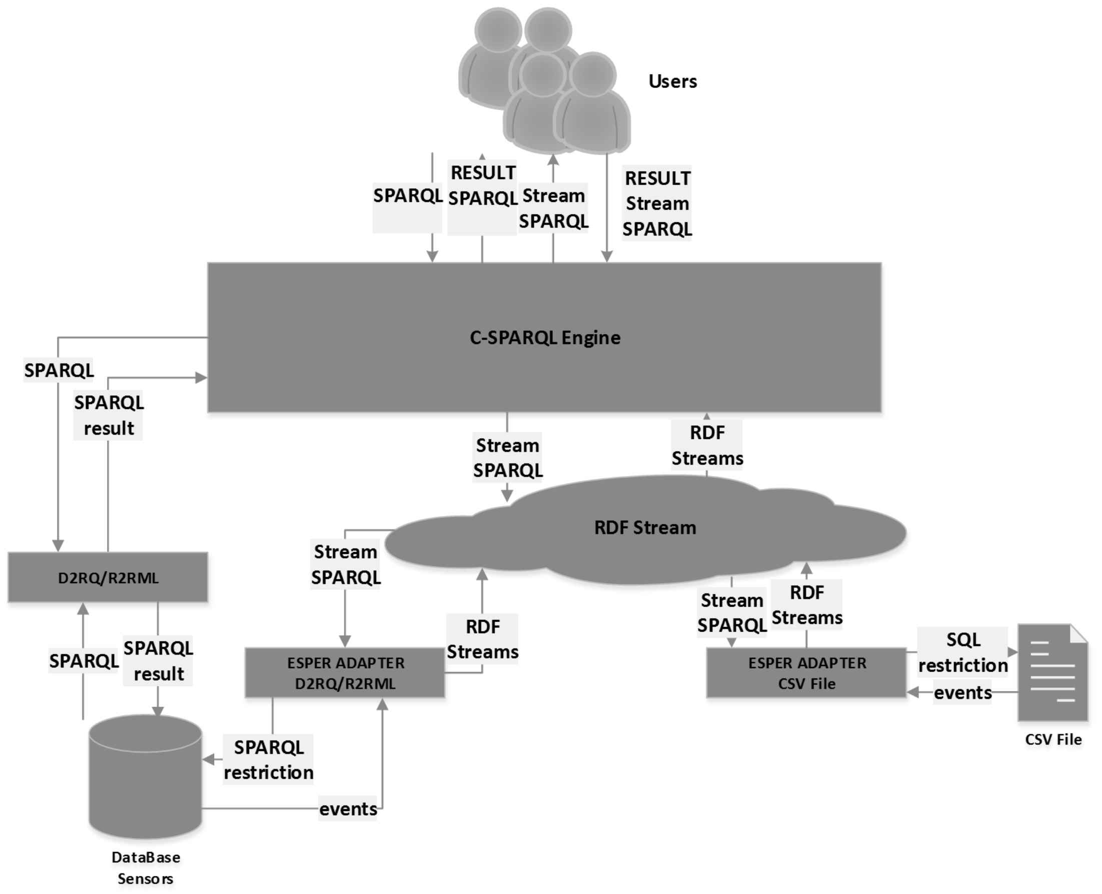


Figure 16 "Stream SPARQL Architecture based on C-SPARQL"

Based on the C-SPARQL engine, the developed architecture (see *Figure 16*) is centered in the C-SPARQL engine that is capable of listening and reasoning over RDF streams that come from different sources, attending to a defined SPARQL stream query. Then, the process is able to reason over semantic streams or RDF Streams. The RDF Stream (see "*Listing 1*") is a quad formed by a subject ("s"), a property ("p"), an object ("o"), and a timestamp ("τ_i") that indicates when the event is produced.

$$stream \equiv \langle (s, p, o), \tau_i \rangle$$

Listing 1 "RDF Stream Notation"

Based on this stream data structure, the stream SPARQL queries permit the users to access this information semantically. The stream SPARQL is a SPARQL-extended syntax that permit to perform a continuous query over the data and time. This syntax is not standard yet due to the research state of the technology. Then, the basic syntax of C-SPARQL (see *Figure 17*) is divided into three parts: *(i)* *Query From*, that indicates the type of SPARQL query to be performed; *(ii)* *Dataset Clause*, that permit to define the namespace of the stream to be queried and the time window where the query listen for streams; *(iii)* *Where Clause* that defined the needed conditions to guide the query.

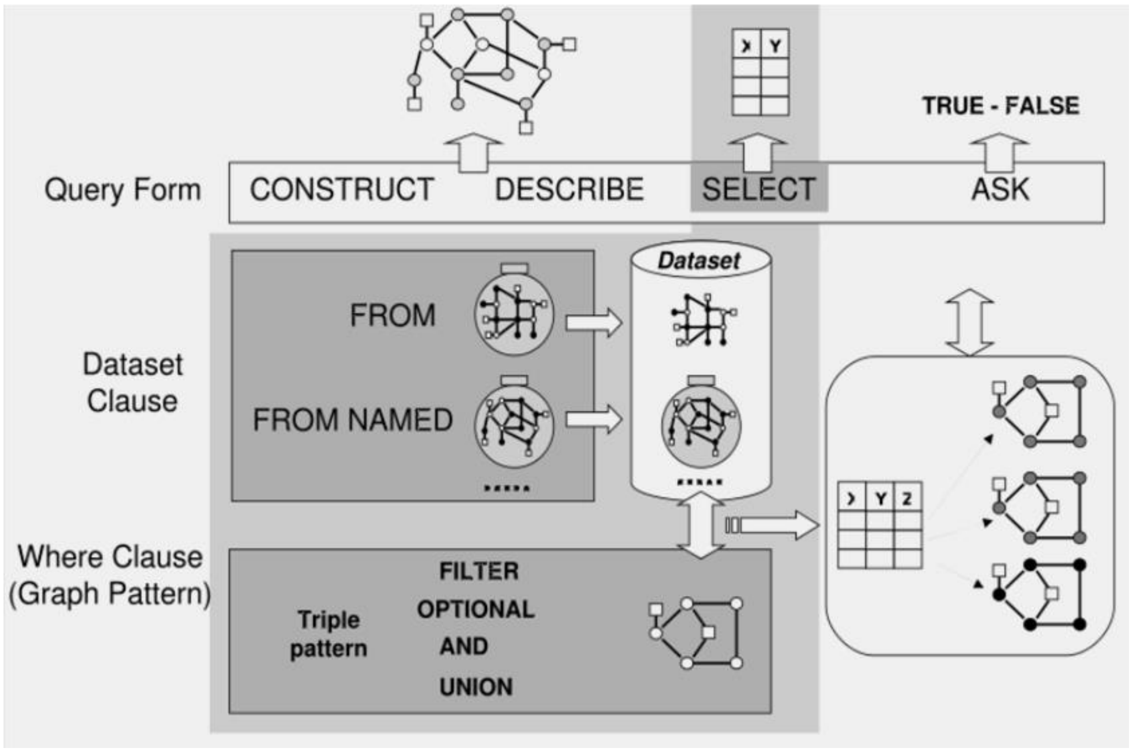


*Figure 17 "*C-SPARQL Syntax*". Source: (Barbieri et al., 2009)*

Based on this extended-SPARQL syntax, the process that C-SPARQL engine follows to execute a query is focused on: *(i)* Initialising the C-SPARQL stream; *(ii)* Connecting the adapters to the engine; *(iii)* registering and listening for results generated by a stream query.

Initialize the C-SPARQL engine means to start and configure the engine (see *Listing 2*). The parameters that can be configured in C-SPARQL are the queue dimension and the enable/disable of time stamp function. In one hand, the queue dimension permit to enable a stream SPARQL injector with the specific dimension. In case the queue dimension is not specified, the injector is not active. On the other hand, the enable/disable parameter for the time stamp function is a Boolean that activates or deactivates the usage of time stamp in the streams. In the developed case, an initialization using the timestamp function is used in order to mark the incoming data streams with a timestamp to reason over them.

```
CsparqlEngine engine = new CsparqlEngineImpl();
engine.initialize(true);
```

Listing 2 "Initialization of C-SPARQL engine"

Once the system is initialized, the **connection of the adapters with the engine** is focused on initializing the adapter's connection and starting them in order to acquire semantic stream for the sources. In the architecture, a **database adapter** and a **CSV file adapter** have been implemented. The main aim of these modules of the architecture are focused on transforming the original information into RDF-Streams according to the developed semantic domain model (see "*Semantic Structure Definition*" section). As can be appreciated during the stream informational querying, the information is stored in multiple systems that has to be collected and manipulated as unique informational store. This aspect is an advantage in environments where the response time is crucial. Therefore, the initialization of the adapters into the system is done by the definition of a functions capable of connecting to the data source, extract it information and transform this information into RDF stream information (see *Listing 3*). Mainly, these functions requires as a parameters: the connection to the CSV or data base, the URI corresponding the semantic domain name space, a restriction over the source data that permit to perform an initial filter, and the stream id to identify the streams into the architecture.

```
esperAdaper adapter= new esperAdaper();
tg = adapter.getEsperStreams(“turbi3000_ea51_280514_exemple.dat”,
                            “http://www.udl.griho.cat/semanticsensorweb”,
                             “select * from TurbidityCinca.win:length(100)”,
                            “TurbidityCinca”,
                            configuration);
…
RDBadapter rdbAdapter= new RDBadapter();
anotherTg= rdbAdapter.getMorphStreams(“ResultR2RML.csv”,
                                   “http://www.udl.griho.cat/semanticsensorweb”,
                                   “select * from SensorObservation.win:length(100)”,
                                   “SensorObservation”
                                   configuration,
                                   “R2RQConfiguration.conf”);
```

Listing 3 “Connection of the adapters with the engine”

Once the connections with the adapters have been established, the adapters have to be **registered into the C-SPARQL** (see *Listing 4*). At this moment, C-SPARQL engine is able of receiving information from the different data sources.

```
engine.registerStream(tg);
engine.registerStream(anotherTg);
```

Listing 4 “Registration of the adapters in the engine”

Once configured and registered the adapters into the architecture, stream SPARQL queries are ready to be listened and executed. The process of listening stream SPARQL strictly depends on the stream **SPARQL query definition** and registration. The definition of the stream SPARQL query relies on the C-SPARQL notation. Based on the stream query definition, the **stream query registration** into the system (see *Listing 5*) means that the query has to be associated with an ID in order to able the architecture to perform continuous queries over the adapters. Furthermore, the stream SPARQL has to be associated with specific result formatter in order to export the results to the user.

```
CsparqlQueryResultProxy c1 = null;
c1 = engine.registerQuery(query);
c1.addObserver(new
cat.udl.eps.griho.ssw.knowledgebase.adapters.Console.ConsoleFormatter());
```

Listing 5 "Stream query registration"

Hereafter, the C-SPARQL engine listen for all the RDF streams that accomplish the time window defined inside the stream query (see *Listing 6*). Therefore, at same time as the RDF-Stream is generated into the system, the C-SPARQL engine return the result to the user by taking into account the conditions under the "*WHERE*" clause and the time stamp defined.

$$\langle (s_i, p_i, o_i), \tau_i \rangle$$

$$\langle (s_{i+1}, p_{i+1}, o_{i+1}), \tau_{i+1} \rangle \; \forall \, i \leq T$$

Listing 6 "C-STREAM SPARQL streams read in the query"

Secondly, plain SPARQL queries can be executed over databases in order to collect semantic information. To perform this kind of queries, D2RQ/R2RML technology permit to generate a bridge between the semantic model and the specific database. Hereafter, these tools permit to abstract the tables, corresponding attributes, and the relation between tables into entities, data properties and object properties according to a specific mapping file (more detailed in "*Database to RDF Stream Adapter*" Section).

As a conclusion, the developed stream architecture permit the users to perform an online reasoning over the generated data. This is a highlight aspect in tools where detection and response time is crucial. Furthermore, a static SPARQL has been defined in order to perform off-line analysis over stored data that permit to identify initial patterns and rules that can help in the anomalous behavioural detection.

Construction of the adapters

As mentioned in the previous section, the main weakness of the C-SPARQL engine is the configuration of the adapters that feed with RDF-Stream the engine. Then, the interaction between the streaming querying engine and the data sources (databases, webs and CSV files) requires for the development of an adapter. In the present case, the developed adapters correspond with a "*Database to RDF Stream*" and "*CSV file to RDF Stream*". On the one hand, "*Database to RDF Stream*" (see "*Database to RDF Stream Adapter*" section) adapter has been used to obtain SQL-structured information and include this information into the platform as RDF Streams. Then, this adapter uses semantic technology that permit to abstract a data base into a semantic layer, and then, SPARQL queries can be performed in order to obtain Structured Query Language(SQL) data as an ontology query. This technology corresponds to the usage of D2RQ or R2RML libraries. In the present research, R2RML has been included into the architecture due to the faster mapping that provides between the data base structure and the semantic resource alignments.

On the other hand, "*CSV file to RDF Stream*" (see "*CSV file to RDF Stream Adapter*" section) permit the architecture to access CSV information by the usage of Esper library. The Esper library permits the adapter to read a CSV file and transform the contained content into "events" that in turn, the contained information is converted into RDF Streams.

Database to RDF Stream Adapter

The main aim of the "*Database to RDF Stream*" adapter is to transform the RDF or SQL-structured information into RDF Stream information. Then, to generate the RDF Streams, a

```
map:foi_off__link a d2rq:PropertyBridge;
        d2rq:belongsToClassMap map:feature_of_interest;
        d2rq:property semanticsensorweb:hasOffering;
        d2rq:refersToClassMap map:offering;
        d2rq:join "foi_off.feature_of_interest_id =>
feature_of_interest.feature_of_interest_id";
        d2rq:join "foi_off.offering_id => offering.offering_id";
```

Listing 7 "D2RQ mapping"

domain model and a querying technology to obtain data from data base, are needed. The domain model is known by the definition of the semantic knowledge base (see "*Semantic Structure Definition*" section). Moreover, the query technology to obtain information from the database has been implemented by the usage of techniques such as D2RQ or R2RML. Both techniques permit to create a mapping between the SQL-structure of the database and the domain knowledge base model. This transformation is defined using a file named mapping file. In both cases, the mapping file is a pseudo-RDF file that includes the rules to be followed in order to convert the specific database into semantic resources. At this moment, the primary difference between D2RQ and R2RML is in the way to represent the transformation. In D2RQ, entities are related with tables and the relation among tables are translated as an object properties (see *Listing 7*). Then, the properties are translated as a joins between the involved tables.

In case of R2RML, the translation is performed at SQL-query level, then an entity is formed by a SQL query and it properties are associated with the attributes defined in the SQL-query (see Listing 8). This distinction between transformation strategies makes a big the difference at time of converting SQL sentences into semantic resources. That means, time elapsed in the transformation of nested joins in different queries is bigger than the time elapsed

```
<#DeptTableViewFeatureOfInterest> rr:sqlQuery """
        SELECT feature_of_interest.feature_of_interest_id,
       feature_of_interest.feature_of_interest_name,
       feature_of_interest.feature_of_interest_description,
       feature_of_interest.feature_type,
       feature_of_interest.schema_link,
          st_astext(feature_of_interest.geom) AS geometry,
       foi_off.offering_id AS offering_id,
       proc_foi.procedure_id AS procedure_id
        FROM feature_of_interest, foi_off,proc_foi
        WHERE (feature_of_interest.feature_of_interest_id = foi_off.feature_of_interest_id)
        AND (feature_of_interest.feature_of_interest_id = proc_foi.feature_of_interest_id);
""".
<#TriplesMapFeatureOfInterest>
        a rr:TriplesMap;
        rr:logicalTable <#DeptTableViewFeatureOfInterest>;
…
.
```

Listing 8 "R2RML mapping"

in the transformation of nested joins in a single query.

This difference can be noticed during the transformation of OGC-SOS database into semantic resources. In the *Table 3*, conversion time for mentioned approaches has been analysed for several queries. These queries correspond to a simple query and a complex query. The simple query has been defined as gathering all information related to the observations available in the system (see *Listing 10*). The execution of the simple query reveals that time elapsed in D2RQ engine (~14000 ms) is notably higher than the provided for the same query for R2RML (~3000 ms).

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX base:<http://www.udl.griho.cat/semanticsensorweb#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?obs
WHERE {
    obs a ssn:Observation .
}
```

Listing 10 "Simple Query SPARQL code"

In the complex query execution, the query is defined as getting all complete information regarding with an observation. Then in this query, an interaction with the "*Phenomenon*" and the "*FeatureOfInterest*" tables/entities are needed for each observation (see *Listing 9*).

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX base:<http://www.udl.griho.cat/semanticsensorweb#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?obs ?obsName ?numValue ?datTime ?foi ?phen
WHERE {
    ?obs a ssn:Observation .
    ?obs base:hasID ?obsName
    ?obs base:hasNumericValue ?numValue .
    ?obs base:hasDateTime ?datTime .
    ?obs base:isObservationOf ?foi .
    ?obs ssn:observedProperty ?phen .
}
```

Listing 9 "Complex Query SPARQL code"

As a result, the R2RML (~8800) for this complex query makes notorious the difference against D2RQ (916,939 ms) when multiple tables are involved in the query. In this case, an exponential increment of time can be noticed due to the execution of nested joins for each observation generated by D2RQ.

	Query Simple (20160 triples)	**Query Complex (20160 triples)**
D2RQ	~ 14,000 ms	~916,939 ms
R2RML	~3,000 ms	~8,800ms

Table 3 "RDBtoRDF engines comparison"

This time difference in time execution has permitted to select R2RML as a SQL to RDF/OWL transformation engine. Based on this tool, the adapter architecture (see *Figure 18*) is focused on using this engine in order to concrete the generation of RDF Streams. For this purpose, an SPARQL query has been defined in order to gather all sensor events from the data base (see *Listing 11*).

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX base: <http://www.udl.griho.cat/semanticsensorweb#>
SELECT DISTINCT ?obs ?obsName ?dt ?foi ?phen ?numVal
WHERE {"
    ?obs a ssn:Observation .
    ?obs base:hasID ?obsName . "
    ?obs base:hasDateTime ?dt
    ?obs base:isObservationOf ?foi
    ?obs ssn:observedProperty ?phen ."
    ?obs base:hasNumericValue ?numVal
}
```

Listing 11 "Defined SPARQL query to gather sensor events"

Based on this SPARQL query, the R2RML engine execution and the consequent RDF Stream transformation is performed in the "*RDFStreamTransformator*". The "*RDFStreamTransformator*" is a configured Esper module capable of sending the SPARQL query to the R2RML engine by using a process call (see *Listing 12*).

```
String command= "java -cp C:\D2RQCSPARQL-all-1.0.jar;
cat.udl.eps.griho.ssw.knowledgebase.endpoint.StreamQuery
\""+configFilePath+""\ \""+sparqlQuery+"\"";
Process process = Runtime.getRuntime().exec(command);
```

Listing 12 "R2RML process execution"

During this process call, the R2RML engine executes the SPARQL query by transforming this SPARQL into SQL sentences helped by the defined mapping file (see *Listing 13*).

```
SELECT v_69716.observation_id AS "var_obs",
  326570829 AS "mappingid_obs",
  'http://www.udl.griho.cat/semanticsensorweb#hasID' AS "uri_hasID2087822614",
  'http://www.udl.griho.cat/semanticsensorweb#hasDateTime' AS "uri_hasDateTime475286966",
  'http://www.udl.griho.cat/semanticsensorweb#isObservationOf' AS
"uri_isObservationOf1495144718",
  'http://purl.oclc.org/NET/ssnx/ssn#observedProperty' AS "uri_observedProperty276810356",
  'http://www.udl.griho.cat/semanticsensorweb#hasNumericValue' AS
"uri_hasNumericValue757108621",
  v_69716.observation_id AS "var_obsName",
  474368464 AS "mappingid_obsName",
  v_69716.time AS "var_dt",
  1251476784 AS "mappingid_dt",
  v_69716.feature_of_interest_id AS "var_foi",
  1115469970 AS "mappingid_foi",
  v_69716.phenomenon_id AS "var_phen",
  171547480 AS "mappingid_phen",
  v_69716.numeric_value AS "var_numVal",
  1275442086 AS "mappingid_numVal"
FROM observation v_69716
 WHERE v_69716.time IS NOT NULL AND v_69716.feature_of_interest_id IS NOT NULL AND
  v_69716.phenomenon_id IS NOT NULL AND v_69716.numeric_value IS NOT NULL AND
  v_69716.observation_id IS NOT NULL

SELECT  DISTINCT v_69716.numeric_value AS "numVal",
  v_69716.phenomenon_id AS "phen",
  v_69716.observation_id AS "obsName",
  1251476784 AS "mappingid_dt",
  v_69716.feature_of_interest_id AS "foi",
  1275442086 AS "mappingid_numVal",
  326570829 AS "mappingid_obs",
  v_69716.time AS "dt",
  474368464 AS "mappingid_obsName",
  v_69716.observation_id AS "obs",
  1115469970 AS "mappingid_foi",171547480 AS "mappingid_phen"
FROM observation v_69716
WHERE v_69716.time IS NOT NULL AND v_69716.feature_of_interest_id IS NOT NULL AND
  v_69716.phenomenon_id IS NOT NULL AND
 v_69716.numeric_value IS NOT NULL AND
  v_69716.observation_id IS NOT NULL

…
```

Listing 13 "R2RML to SQL transformation"

Once R2RML returns the asked values, the "*RDFStreamTransformation*" generates the RDFStreams based on the information contained in the generated R2RML output stream. This transformation is done by using an Esper module capable of reading the R2RML output file as an input file and converting the contained information into RDF stream triples (see *Listing 14*.).

```
EPServiceProvider epService = EPServiceProviderManager.getDefaultProvider(configuration);
stmt = epService.getEPAdministrator().createEPL(esperRestricton);

this.tg = new CEPDataBaseListener(OntIRI, stmt, epService);

CSVInputAdapterSpec spec = new CSVInputAdapterSpec(new AdapterInputSource(CSVFileName),
eventName);
inputAdapter = new CSVInputAdapter(epService, spec);
```

Listing 14 "RDF Stream transformation from R2RML"

The conversion towards RDF streams is done under the class "*CEPDataBAseListener*" in charge of waiting for data streams and convert the information into RDF streams by following the *Listing 15* triples.

```
RdfQuadruple q = new RdfQuadruple((String) eventBeans[0].get("obs"),
                "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
                "http://purl.oclc.org/NET/ssnx/ssn#Observation",
                System.currentTimeMillis());
put(q);

q = new RdfQuadruple((String) eventBeans[0].get("obs"),
                super.getIRI()+"#hasName",
                super.getIRI() + "#"+eventBeans[0].get("obsName"),
                System.currentTimeMillis());
put(q);

q = new RdfQuadruple((String) eventBeans[0].get("obs"),
                super.getIRI()+"#hasDateTime",
                (String)eventBeans[0].get("dt"),
                System.currentTimeMillis());

put(q);

        q = new RdfQuadruple((String) eventBeans[0].get("obs"),
                super.getIRI()+"#hasNumericValue",
                eventBeans[0].get("numVal")+"^^http://www.w3.org/2001/XMLSchema#decimal",
                System.currentTimeMillis());

put(q);

…
```

Listing 15 "Example of RDF streams generation from R2RML information"

During the generation of RDF streams triples, streams are sent to the C-SPARQL engine using the "*put*" operation defined in the "*RdfStream*" class of C-SPARQL engine. In other words, this function permit to link the configured Esper engine in this adapter with the internal Esper adapter of the C-SPARQL engine. Therefore, the adapter is able to accomplish the objective of gather information from a database without replicating the information into a triple-quad store and the bottlenecks and maintenance cost that it provides.

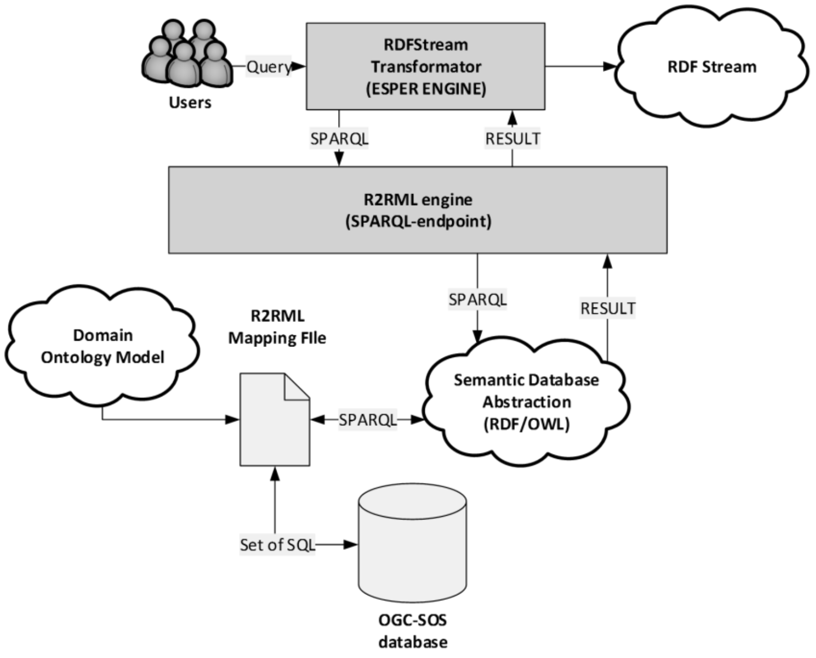


Figure 18 "Database Adapter Architecture"

CSV file to RDF Stream Adapter

Similarly as previous adapter, the "*CSV to RDFStream*" adapter main purpose is to get CSV file (from location or URL), read the document and transform the fields of the CSV into RDFStreams. To perform this process, the defined architecture (see *Figure 19*) is based on an "*RDFStreamTransformator*" based on Esper engine capable of transforming the incoming sensor information into streams.

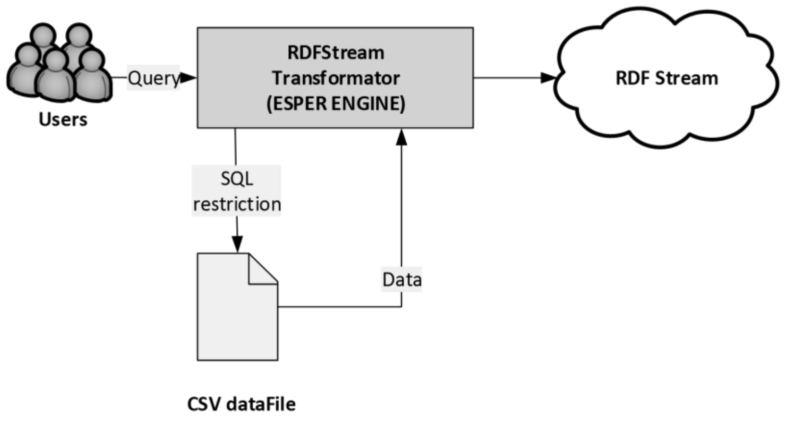


Figure 19 "CSV Adapter Architecture"

This Esper engine is capable of dealing with CSV files in a stream mode by using the Esper-CSV adapter (see *Listing 16*). This adapter needs for a CSV input file, SQL-CSV restriction query and a configuration field to indicate Esper the information contained in the CSV.

```
EPServiceProvider epService = EPServiceProviderManager.getDefaultProvider(configuration);
stmt = epService.getEPAdministrator().createEPL(esperRestricton);

tg = new CEPListener(OntIRI, stmt, epService);

CSVInputAdapterSpec spec = new CSVInputAdapterSpec(new AdapterInputSource(CSVFileName),
eventName);
inputAdapter = new CSVInputAdapter(epService, spec);
```

Listing 16 "RDF Stream transformation from CSV"

Then, the "*RDFStreamTransformator*" applies the SQL restriction to the CSV file and gets the requested information as Esper events ("*CEPListener*" class). Once the Esper events are received (based on the CSV information), the events are transformed into RDF streams based on the domain ontology (see *Listing 17*).

```
RdfQuadruple q = new RdfQuadruple(super.getIRI() + "#turbidityObservation"+
eventBeans[0].get("RECORD")+"Turb_Avg",
                "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
                "http://purl.oclc.org/NET/ssnx/ssn#Observation",
                System.currentTimeMillis());

put(q);

q = new RdfQuadruple(super.getIRI() + "#turbidityObservation"+
eventBeans[0].get("RECORD")+"Turb_Avg",
                super.getIRI()+"#hasNumericValue",
eventBeans[0].get("Turb_NTU_Avg")+"^^http://www.w3.org/2001/XMLSchema#decimal",
                System.currentTimeMillis());

put(q);

q = new RdfQuadruple(super.getIRI() + "#turbidityObservation"+
eventBeans[0].get("RECORD")+"Turb_Avg",
                super.getIRI()+"#hasNumericValue",
eventBeans[0].get("TurbNTUSD_Avg")+"^^http://www.w3.org/2001/XMLSchema#decimal",
                System.currentTimeMillis());
put(q);
```

Listing 17 "Example of RDF streams generation from CSV information"

Similarly as the "*Database to RDF Stream Adapter*" section, the RDF Streams are sent to the C-SPARQL engine using the "*put*" function derived from the "*RdfStream*" that is the parent class of "*CEPListener*". Therefore, both the CSV and the internal C-SPARQL Esper engine are thus connected and semantic streams are registered in order to be queried.

As a conclusion, the "*CSV to RDF Stream*" adapter permit to get, load and transform a CSV file into RDF events that can be read by the C-SPARQL architecture. This adapter has been based on the Esper engine that links the developed adapter Esper with the C-SPARQL Esper engine to manage semantic streams. Then, the CSV file events have been transformed into semantic events giving the capability of reasoning over this information.

Initial results of the architecture

Once the information has been built, this architecture has initially tested by implementing different streaming SPARQL queries over the available information. Then, a testing scenario (see

Figure 20) has been defined and implemented. This test scenario gathers information from an installed OGC-SOS server into a virtual machine, and CSV file extracted from the observations performed over the Cinca River (Lleida, Spain) during the period of time between *2014/04/29* and *2014/05/28*.

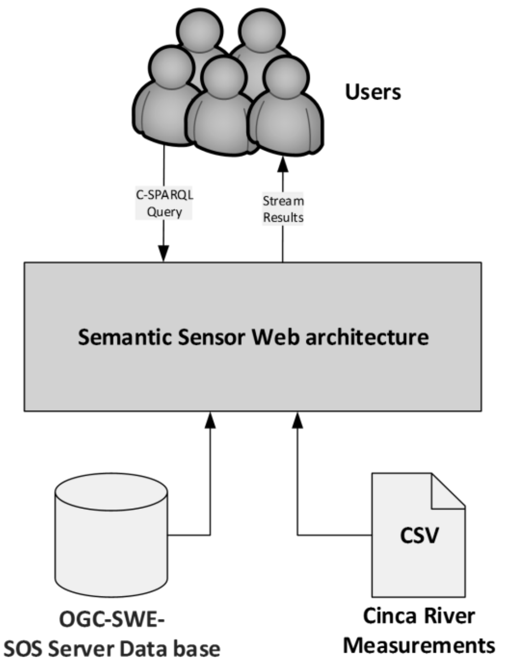


Figure 20 "Testing Scenario architecture"

Over this scenario, different queries have been executed in order to evaluate the performance of the system. Then, the first performed stream SPARQL query (see *Listing 18*) is aimed at getting all the available observations from sensors defined in both data sources. In this mentioned query, the first sentence ("*REGISTER QUERY*" tag) permit to identify the query in the

```
REGISTER QUERY GetObservations AS
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX base:<http://www.udl.griho.cat/semanticsensorweb#>\
SELECT ?obs
FROM STREAM <http://www.udl.griho.cat/semanticsensorweb> [RANGE 5s STEP 1s]
WHERE {
        ?obs a ssn:Observation .
      }
```

Listing 18 "Stream query for getting all the observations"

C-SPARQL engine after registering the query. Another special tag is under the "*FROM*" clause where the watched URI, the observed window ("*RANGE*") and the step between windows ("*STEP*"), are defined.

On the one hand, the observed window represent the observed time interval. On the other hand, the step between windows represent the time difference between continuous queries over time. In this case, the range is defined as a *5*s and the step as *1*s. Thus, this values represent that each observed window has a length of *5*s and the observations are done every second.

As a result (see *Listing 19*), the system returns all the observations (URIs) that are happening inside a window of *5*s every second. Thanks to the window threshold, several observations are repeated during some steps.

```
-------3 results at SystemTime=[1409313441170]--------
http://www.udl.griho.cat/semanticsensorweb#turbidityObservation1.0Turb_Min
http://www.udl.griho.cat/semanticsensorweb#733468
http://www.udl.griho.cat/semanticsensorweb#turbidityObservation1.0Turb_Max
-------8 results at SystemTime=[1409313441795]--------
http://www.udl.griho.cat/semanticsensorweb#turbidityObservation2.0Turb_Min
http://www.udl.griho.cat/semanticsensorweb#733412
http://www.udl.griho.cat/semanticsensorweb#turbidityObservation2.0Turb_Max
http://www.udl.griho.cat/semanticsensorweb#733482
http://www.udl.griho.cat/semanticsensorweb#turbidityObservation2.0Turb_Avg
http://www.udl.griho.cat/semanticsensorweb#turbidityObservation1.0Turb_Min
http://www.udl.griho.cat/semanticsensorweb#733468
http://www.udl.griho.cat/semanticsensorweb#turbidityObservation1.0Turb_Max
```

Listing 19 "Result of the stream SQPARQL that gets all the observations"

The second query is aimed at selecting all observations whose numeric values where less than *0* (see *Listing 21*). This streaming query can serve to see which sensors are gathering wrong information in reference to the environment. As can be noticed, the structure of the SPARQL is similar as the query exposed above. The only difference is the incorporation of the "*FILTER*" clause that permit to establish an acceptance thresholds over the semantic instances.

```
REGISTER QUERY AllInstances AS
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX base:<http://www.udl.griho.cat/semanticsensorweb#>
SELECT ?obs ?val
FROM STREAM <http://www.udl.griho.cat/semanticsensorweb> [RANGE 5s STEP 1s]
WHERE {?obs a ssn:Observation .
       ?obs base:hasNumericValue ?val .
       FILTER (?val < 0)}
```

Listing 21 “Stream query for getting all the observations”

As a result, the stream query permit to know by each time windows whose sensors are generating anomalies in the system in reference with it negative values (see *Listing 20*).

```
-------1 results at SystemTime=[1409317443877]--------
http://www.udl.griho.cat/semanticsensorweb#733440"-12.8"^^http://www.w3.org/2001/XMLSchema#decimal
-------3 results at SystemTime=[1409317444676]--------
http://www.udl.griho.cat/semanticsensorweb#733468"-12.8"^^http://www.w3.org/2001/XMLSchema#decimal
http://www.udl.griho.cat/semanticsensorweb#733454"-12.8"^^http://www.w3.org/2001/XMLSchema#decimal
http://www.udl.griho.cat/semanticsensorweb#733440"-12.8"^^http://www.w3.org/2001/XMLSchema#decimal
```

Listing 20 “Result of the stream SQPARQL that gets all the observations with negative numeric value”

As a conclusion of the results, the developed architecture permit to query information regarding multiple data sources in a stream mode. That means, by making continuous queries over time in order to see how the sensor data is evolving. Furthermore, this architecture also permit to query information with the main aim of selecting those attributes and values that permit to identify anomalies in the system.

CONCLUSIONS AND FUTURE WORK

The semantic sensor web is an innovative and progressive approach that is in continuous development and evolvement. Throughout this document, main current state of this approach has been depicted. This current state of the semantic sensor web approaches covers the standard definition of a sensor web architecture under the umbrella of the OGC-SWE. This approach permits to collect, store, manage and publish sensor information through a web services based on a service oriented architecture. Furthermore, standard XML languages for sensor data sharing has been defined throughout the OGC-SWE approach. As an example, the OGC-SWE provides XML schemas for sharing water observations (WaterML), general sensor information (SensorML) and smart city information (CityML) among others. So, the OGC-SWE is synonym of standardisation and also cross-domain data sensor collector. However, current weakness of the systems lies in the collection and storage of huge amount of data in databases and non-relational data bases. This last aspect is not a sustainable strategy that rewards data quantity against quality.

In order to reward the quality, semantic sensor web concept is focused on generating and reusing effective knowledge in a cross-domain environments. To achieve this aspect, semantic structures to describe sensor nature in different domains have been developed. The most know sensor domain knowledge base is the one developed by the W3C called SSN-XG. By the use of semantic structures the achievable benefits can be measured in terms of *(i)* semantic interoperability between systems by the development of a common language that permit to share and understand the information; *(ii)* fuse heterogeneous data sources by abstracting the informational nature and storage; and *(iii)* generate new information by the understanding of the gathered information.

Other challenges and features of the semantic sensor web approaches are maintenance of data quality and security; data integrity, data fusion and abstraction of the concepts across different domains; reasoning over geo-spatial location of sensors; mechanisms to assure data provenance; and the generation of a general application that permit the creation of domain-specific ones.

Based on this starting point, the presented approach has been focused on the challenges of generating **effective knowledge and the automatic data fusion and integration through the construction of an architecture capable of collecting related-sensor information from different data sources, abstracting this information by the definition of a semantic layer and reason and sharing collected information by using semantic end-points (C-SPARQL and SPARQL end-points)**. Based on this definition of the architecture, the main highlights can be summarised as *(i)* an **architecture able of fusing sensor information with other kind of information** into a single platform, and (ii) **sustainable platform that uses streamed information without replicating sensor information** from other sources.

In spite of the direct benefits of this architecture, this developed paradigm is a small part of a bigger architectural challenge. So, the proposed research lines can be focused on enhancing the application by *(i)* the application of data mining, stream data mining and big data techniques towards generating rules that can enhance data reasoning and knowledge generation; *(ii)* developing a set of adapters in order to increase the informational coverage into the architecture; *(iii)* deploying the architecture in a web service architecture based on REST services and JSON-LD language for data sharing and understanding by webs and applications; and (iv) adding into the architecture a wisdom layer, that means, an intelligent interaction with users that permit to introduce human-reasoning and experience into the system.

BIBLIOGRAPHY

Aberer, K., Hauswirth, M., & Salehi, A. (2006). Global sensor networks, (5005). Retrieved from http://infoscience.epfl.ch/record/64020

Alam, S., Chowdhury, M. M. ., & Noll, J. (2010). SenaaS:An Event-driven Sensor Virtualization Approach for Internet of Things Cloud. *Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on*, 1–6.

Amato, F., Casola, V., Gaglione, A., & Mazzeo, A. (2011). A semantic enriched data model for sensor network interoperability. *Simulation Modelling Practice and Theory*, *19*(8), 1745–1757. doi:10.1016/j.simpat.2010.09.010

Anicic, D., & Fodor, P. (2011). EP-SPARQL: a unified language for event processing and stream reasoning. *Proceedings of the 20th …*. Retrieved from http://dl.acm.org/citation.cfm?id=1963495

Barbieri, D., Braga, D., & Ceri, S. (2009). C-SPARQL: SPARQL for continuous querying. *Proceedings of the 18th …*, (c), 1061–1062. Retrieved from http://dl.acm.org/citation.cfm?id=1526856

Barnaghi, P., & Wang, W. (2012). Semantics for the Internet of Things: early progress and back to the future. *International Journal on …*. Retrieved from http://www.igi-global.com/article/content/70584

Berjon, R., Faulkner, S., Leithead, T., Navara, E. D., O'Connor, E., & Pfeiffer, S. (2014). HTML5: A vocabulary and associated APIs for HTML and XHTML. *W3C*. Retrieved from http://www.w3.org/TR/html5/

Bizer, C., & Berlin, F. U. (2009). Linked Data -The Story So Far.

Boncz, P., & Pham, M.-D. (2013). BSBM V3.1 Results. *CWI*.

Bröring, A., Echterhoff, J., Jirka, S., Simonis, I., Everding, T., Stasch, C., … Lemmens, R. (2011). *New generation Sensor Web Enablement. Sensors (Basel, Switzerland)* (Vol. 11, pp. 2652–99). doi:10.3390/s110302652

Bröring, A., Janowicz, K., Stasch, C., & Kuhn, W. (2009). Semantic challenges for sensor plug and play. *Web and Wireless …*. Retrieved from http://iospress.metapress.com/index/F52644V462H28KX4.pdf

Broring, A., & Maué, P. (2012). Semantic mediation on the Sensor Web. *... and Remote Sensing ...*, 1–4. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6350717

Brunetti, J. M., García, R., & Auer, S. (2013). From Overview to Facets and Pivoting for Interactive Exploration of Semantic Web Data. *International Journal on Semantic Web and Information Systems*, *9*(1), 1–20. doi:10.4018/jswis.2013010101

Calbimonte, J., Corcho, O., & Gray, A. (2010). Enabling ontology-based access to streaming data sources. *The Semantic Web–ISWC 2010*, (September), 1–16. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-17746-0_7

Calbimonte, J.-P., Jeung, H., Corcho, O., & Aberer, K. (2012). Enabling Query Technologies for the Semantic Sensor Web. *International Journal on Semantic Web and Information Systems*, *8*(1), 43–63. doi:10.4018/jswis.2012010103

Chen, C., & Helal, S. (2008). Sifting through the jungle of sensor standards. *Pervasive Computing, IEEE*, 84–88. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4653477

Compton, M., Barnaghi, P., Bermudez, L., García-Castro, R., Corcho, O., Cox, S., … Taylor, K. (2012). The SSN ontology of the W3C semantic sensor network incubator group. *Web*

Semantics: Science, Services and Agents on the World Wide Web, *17*, 25–32. doi:10.1016/j.websem.2012.05.003

Compton, M., Henson, C., & Neuhaus, H. (2009). A Survey of the Semantic Specification of Sensors. *SSN*. Retrieved from http://www.academia.edu/download/30227318/compton2009_-_a_survey_of_the_semantic_specifications_of_sensor.pdf

Corchero, A., Domingo, X., & García, R. (2013). Semantic sensor web data exploration and visualization for intelligent decision support. *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics - WIMS '13*, 1. doi:10.1145/2479787.2479831

Corcho, O., & García-Castro, R. (2010). Five challenges for the semantic sensor web. *Semantic Web*, *1*, 121–125. doi:10.3233/SW-2010-0005

Cudré-Mauroux, P., & Enchev, I. (2013). Nosql databases for rdf: An empirical evaluation. *The Semantic Web– …*. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-41338-4_20

ECHONET. (2013). *Energy Conservation and Homecare Network* (p. 10).

EDDL. (2007). *EDDL Technical Description* (pp. 1–17).

Esswein, S., Goasguen, S., Post, C., Hallstrom, J., White, D., & Eidson, G. (2012). Towards Ontology-based Data Quality Inference in Large-Scale Sensor Networks. *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, 898–903. doi:10.1109/CCGrid.2012.143

Exhibit. (2012). *Exhibit 3.0 Documentation* (p. 67).

García, R., & Brunetti, J. (2011). Publishing and interacting with linked data. *Proceedings of the …*. Retrieved from http://dl.acm.org/citation.cfm?id=1988710

García, R., Gimeno, J., & Perdrix, F. (2008). The rhizomer semantic content management system. *... Information Systems for the …*. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-87781-3_42

Gray, A. J. G., Calbimonte, J., Page, K., Sadler, J., Galpin, I., Fernandes, A. A. A., … Martinez, K. (2011). A semantically enabled service architecture for mashups over streaming and stored data. *The Semanic Web: …*. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-21064-8_21

Hasan, S., & Curry, E. (2011). Toward situation awareness for the semantic sensor web: Complex event processing with dynamic linked data enrichment. *Semantic Sensor …*, 60–72. Retrieved from http://research.ict.csiro.au/conferences/ssn/ssn11/semantic_sensor_networks_2011.pdf#page=66

Henson, C., & Pschorr, J. (2009). SemSOS: Semantic sensor observation service. *... and Systems, 2009. …*, 44–53. doi:10.1109/CTS.2009.5067461

Henson, C., Sheth, A., & Thirunarayan, K. (2012). Semantic perception: Converting sensory observations to abstractions. *Internet Computing, IEEE*, *16*(2), 26–34. doi:10.1109/MIC.2012.20

Henson, C., Thirunarayan, K., & Sheth, A. (2011). An ontological approach to focusing attention and enhancing machine perception on the Web. *Applied Ontology*, *45435*, 1–43. Retrieved from http://iospress.metapress.com/index/17X610N66577N55W.pdf

Herman, I., Adida, B., Sporny, M., & Birbeck, M. (2013). RDFa 1.1 Primer - Second Edition. *W3C*. Retrieved from http://www.w3.org/TR/xhtml-rdfa-primer/

Hobbs, J. R., & Pan, F. (2006). Time Ontology in OWL. Retrieved from http://www.w3.org/TR/owl-time/

Hodgson, R., Keller, P. J., Hodges, J., & Spivak, J. (2014). QUDT - Quantities, Units, Dimensions and Data Types Ontologies. *W3C*. Retrieved from http://qudt.org/

Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. *W3C*. Retrieved from http://www.w3.org/Submission/SWRL/

Hu, P., Robinson, R., & Indulska, J. (2007). Sensor standards: Overview and experiences. ... *Sensors, Sensor ...*, (i), 485–490. doi:10.1109/ISSNIP.2007.4496891

IEEE, S. (1999). *IEEE Standard for a Smart Transducer Interface for Sensors and Actuators — Network Capable Application Processor (NCAP) Information Model*.

JPEO-CBD, & SSA. (2008). *Common Chemical , Biological , Radiological , Nuclear (CBRN) Sensor Interface (CCSI) Volume I – Summary and Architecture*.

Jung, J. J. (2011). Semantic preprocessing for mining sensor streams from heterogeneous environments. *Expert Systems with Applications*, *38*(5), 6107–6111. doi:10.1016/j.eswa.2010.11.017

Kifer, M., & Boley, H. (2013). RIF Overview. *W3C*. Retrieved from http://www.w3.org/TR/rif-overview/

Kit, D. (2013). Device Kit Web page.

Klusch, M., Fries, B., & Sycara, K. (2009). OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services. *Web Semantics: Science, Services and …*, (September). Retrieved from http://www.sciencedirect.com/science/article/pii/S1570826808000838

Lee, K. (2007). Sensor standards harmonization-path to achiewving sensor interoperability. *Autotestcon, 2007 IEEE*, 381–388. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4374244

Le-Phuoc, D., & Dao-Tran, M. (2011). A native and adaptive approach for unified processing of linked streams and linked data. *The Semantic Web– …*, *1380*(24761). Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-25073-6_24

Le-Phuoc, D., Dao-Tran, M., & Pham, M. (2012). Linked stream data processing engines: facts and figures. *The Semantic Web– …*, *1380*(24761), 1–12. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-35173-0_20

Le-Phuoc, D., Parreira, J., & Hauswirth, M. (2012). *Linked stream data processing* (pp. 245–289). Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-33158-9_7

Lieberman, J., Singh, R., & Goad, C. (2007). W3C Geospatial Ontologies. *W3C*. Retrieved from http://www.w3.org/2005/Incubator/geo/XGR-geo-ont-20071023/

Lupp, M. (2008). Open Geospatial Consortium. *Encyclopedia of GIS*. Retrieved from http://link.springer.com/content/pdf/10.1007/978-0-387-35973-1_918.pdf

Maué, P., Schade, S., & Duchesne, P. (2009). *Semantic annotations in OGC standards* (pp. 1–50). Retrieved from http://www.citeulike.org/group/7192/article/5207092

Moraru, A., & Mladenić, D. (2012). A framework for semantic enrichment of sensor data. *CIT. Journal of Computing and Information …*, 155–160. doi:10.2498/iti.2012.0381

NIST. (2006). ANSI/IEEE N42.42. *NIST*. Retrieved from http://www.nist.gov/pml/div682/grp04/n42.cfm

Nokia, R. C. (2008). Sensing the World with Mobile Devices. *Nokia Technology Insights Series*, (December).

OGC-SWE. (2011). OGC Reference Model. *Open Geospatial Consortium*.

Phuoc, D. Le, Parreira, J., & Hauswirth, M. (2010). Challenges in Linked Stream Data Processing: A Position Paper. *SSN*. Retrieved from http://people.csail.mit.edu/pcm/tempISWC/workshops/SSN2010/paper10.pdf

Pileggi, S. F. (2010). A Novel Domain Ontology for Sensor Networks. *2010 Second International Conference on Computational Intelligence, Modelling and Simulation*, 443–447. doi:10.1109/CIMSiM.2010.52

Raskin, R., & Pan, M. (2003). Semantic web for earth and environmental terminology (sweet). *Proc. of the Workshop on Semantic Web …*. Retrieved from http://esto.ndc.nasa.gov/conferences/estc2003/papers/A7P2(Raskin).pdf

Raymond, M., Webb, S., & Aymond, P. Ll. (2006). *Emergency Data Exchange Language (EDXL) Abstract :* (pp. 1–42).

Russomanno, D., Kothari, C., & Thomas, O. (2005). Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. *IC-AI*. Retrieved from https://wwwnew.memphis.edu/eece/cas/docs/ica3194.pdf

Sheth, A., & Henson, C. (2012). Semantic Sensor Web.

Sheth, A., & Perry, M. (2008). Traveling the semantic web through space, time, and theme. *Internet Computing, IEEE*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4463390

Sigüenza, Á., Díaz-Pardo, D., & Bernat, J. (2012). Sharing human-generated observations by integrating HMI and the semantic sensor web. *Sensors*, *12*(5), 6307–30. doi:10.3390/s120506307

Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., & Lindström, N. (2014). A JSON-based Serialization for Linked Data. *W3C*. Retrieved from http://www.w3.org/TR/json-ld/

W3C. (2011). Review of Sensor and Observations Ontologies. Retrieved from http://www.w3.org/2005/Incubator/ssn/wiki/Review_of_Sensor_and_Observations_Ontologies#SWAMO

W3C. (2014). RDF Stream Processors Implementation. *Web*. Retrieved from http://www.w3.org/community/rsp/wiki/RDF_Stream_Processors_Implementation#CQELS_and_CQELS-QL

VITA

Aitor Corchero is a computer engineer by the University of Mondragon. Currently, Aitor is a researcher in the Environment and Food department of BDigital Technologic Centre. Mainly, Aitor's work is focused on addressing intelligent tools aligned with the water, energy and food management projects.