SeNsIM-Web: a Service Based Architecture for Sensor Networks Integration

Valentina Casola, Andrea Gaglione, Antonino Mazzeo
Dipartimento di Informatica e Sistemistica
Universita' degli Studi di Napoli, Federico II
Naples, Italy
{*casolav,andrea.gaglione,mazzeo*}*@unina.it*

Abstract

One of the main open issues in the development of applications for sensor network management is the definition of interoperability mechanisms among the several monitoring systems and heterogeneous data. Interesting researches related to integration techniques have taken place, they are primary based on the adoption of sharing data-mechanisms; furthermore in the last years, the Service-Oriented Architecture (SOA) approach has become predominant in many sensor networks projects as it enables the cooperation and interoperability of different sensor platforms at an higher level of abstraction. In this paper we propose an innovative architecture for the interoperability of sensor networks; it is based on web services tecnologies, on the definition of a data model and on SeNsIM, an integration platform. The proposed architecture allows the development of complex applications by integrating heterogeneous data, which can be accessible through services according to standard data format and standard protocols.

1 Introduction

One of the main open issues in the multi-hazard approach to the environmental risk monitoring and management is the integration of heterogeneous data from different sources to manage and elaborate risk mitigation strategies, including emergency management planning. The large diffusion of sensor systems, together with their numerous applications has led to huge heterogeneity in the logic for interfacing and collecting data from these systems.

However in order to automatize the elaboration of such heterogeneous data, specific integration frameworks for accessing different data sources are needed. Such frameworks should be able to access data, sensed by different sensing infrastructures; they should hide the heterogeneity among different sensor systems (in terms of sensor, networking, or middleware technologies) and provide a standard way to access them. Interesting researches related to integration techniques for heterogeneous sensor networks have taken place, but nowadays only few architectures have been proposed. Most of them try to define a common exchange mechanism among different sensor systems in order to facilitate the integration, and provide a software integration layer which allows different sensor systems to collaborate for the same purpose. Very often these solutions are tightly related to proprietary technologies and they are neither scalable nor open. In the last years, the Service-Oriented Architecture (SOA) approach has become a cornerstone in many sensor networks projects. SOA-approach enables the cooperation and interoperability of different sensor platforms as it provides discovery, access and sharing of the services, data, computational and communication resources by the adoption of open standards. New open standards and the adoption of common data model to formally define and represent data knowledge, are the main features of these architectures that enable the definition of cooperative environments. As an example of data model, the OpenGeospatial Consortium provides an XML schema (Sensor Model Language or SensorML [1]) for defining the geometric, dynamic and observational characteristics of sensors and other standards as Observation and Measurement [2] to describe observed phenomenons. The standards: (1) provide general sensor information in support of data discovery, (2) support the processing and analysis of the sensor measurements, (3) support the geolocation of the measured data, (4) provide performance characteristics (e.g. accuracy, threshold, etc.), and (5) archive fundamental properties and assumptions regarding sensor.

The strong advantage of web service architectures is to build information systems enabling the creation of applications by combining loosely coupled and interoperable services without the knowledge of the underlying systems.

On the other hand, the common data model grants interoperability among multi network systems, providing a formal model for the integration of data gathered by different and heterogeneous sources. A formal data model specifies

both network and sensor features and specific features of the sensed data.

Thanks to web services, we can immagine a complex sensor management system as made of many different atomic services that act as *data collection services, integration services and applicative services*; such structure is very useful to improve the accessibility of all data.

In this paper we propose an *integration service*, named SeNsIM-Web as it is an extension of the *SeNsIM* (Sensor Networks Integration and Management) system [3], an integration platform for heterogeneous sensor systems. SeNsIM has been shown being also a valid support for reasoning applications which correlate data coming from different networks that cooperate for a common objective [4, 5]. SeNsIM-Web inherits the SeNsIM data model and its functions which have been re-implemented as web services accessible by a generic user or an application through standard protocols in the World Wide Web.

To prove the applicability of our system, we also propose a graphical client application, which interact with a generic user and make him access such services for an easy retrieval of all networks features and sensed data.

The reminder of the paper is structured as follows: in Section 2 some related works are reported to assess the state of the art of many methodologies and technologies facing interoperability issue among heterogeneous data. In Section 3 we will present an architectural model based on web services to integrate sensor networks. In Sections 4 and 5 we will illustrate the details of SeNsIM-Web, an extension of the SeNsIM platform compliant with web services technologies. Finally in Section 6 some concluding remarks will be given.

2 Related Works

The need to guarantee interoperability among several monitoring systems and integration of heterogeneous data, can be seen from different point of views, the main obstacles may be classified into the following categories: (a) Syntactic Interoperability: how to overcome technical heterogeneity of all sorts, (b) Semantic Interoperability: how to overcome ambiguities and different interpretations, (c) Application Interoperability: how to deliver sustainable and reusable concepts and components independent of the application domain, the technological infrastructure and the organizations, (d) Phenomena Observation: how to evaluate the meaning of the observation from the temporal, spatial and thematic prospective.

Some solutions to specific, but not complete, aspects of interoperability are available in the literature.

The Open Geospatial Consortium (OGC) proposed a suite of specifications, named Sensor Web Enablement (SWE), to model sensor characteristics and services [1, 2]. In particular, the suite includes: (i) Sensor Model Language(Sensor ML), (ii) Observation & Measurement and (iii) Sensor Observation Service. They allow to model sensor and sensor observations, data retrieval mechanism and web services (for access of the sensor data via web); it is possible to specify information as coordinates and timestamps, but they do not allow to state the semantics of the data and the meaning of sensor observations, making difficult the interoperability, the evaluation of the phenomena and the detection of situation awareness.

In [9], the authors define the semantic sensor Web (SSW) 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 SWE, in order to increase interoperability as well as provide contextual information essential for situational knowledge. In particular, this involves annotating sensor data with spatial, temporal, and thematic semantic metadata.

As for integration aspects, some specific interoperable frameworks have been proposed; for example, in [12], Ahn and Chong propose an intelligent bridge for messages exchange between heterogeneous Wireless Sensor Networks (WSNs). They define a general messages exchange mechanism that uses XML as message style and SOAP as transmission protocol. GSN (Global Sensor Networks) [11] facilitates the flexible integration and discovery of sensor networks and hides arbitrary stream data sources behind its virtual sensor abstraction. It enables the user to specify XML-based deployment descriptors in combination with the possibility to integrate sensor network data through plain SQL queries over local and remote sensor data sources.

Other works try to define a middleware integration architecture that enables interoperability between sensor systems. For example, the ESP framework [13] enables sensor systems to be queried without having to deal with the low-level implementation of specific access methods. It provides a mechanism to describe and model sensor systems using ESPml, an XML-based language, by which information regarding the sensor deployment can be specified.

From an architectural point of view, the current leading approach to interoperability is based on Service Oriented Architectures (SOA). In [13] the authors introduce Web Service Resource Framework (WSRF) mechanisms into the core services implementation of the NICTA Open Sensor Web Architecture (NOSA). WSRF expands the functionality of the services to handle simultaneous observational queries to heterogeneous Sensor Networks. Moreover, the GeoICT group at York University [14] have built an OGC SWE compliant Sensor Web infrastructure. They have developed a Sensor Web client capable of visualizing geospatial data, and a set of Web Services called GeoSWIFT [15].

Actually, two main European early warning projects

based on Service Oriented Architectures (SOA) have been proposed; the WIN (Wide Information Network) [17] and ORCHESTRA [18] projects. WIN aims at developing an open and flexible platform to support multi-hazard and risk domains at European level, integrating national and regional data flows in the frame of a Web Service Architecture. It proposes a set of generic services, standard data modeling components that facilitate the deployment on various thematic cases. The WIN Metadata Model is based by large on existing standards (such as Dublin Core, GML and ISO19115) and includes some additional specifications for WIN. On the other hand, the ORCHESTRA architecture adapts the ISO/IEC 10746 Reference Model for Open Distributed Processing to service-oriented architectures. In particular, The web services are implemented using W3C Web services platform and the Geography Mark-up Language (GML).

3 Web Services for Sensor Networks Integration

According to current literature, the reference architectural model should be structured into three different service layers: Data Collection Service Layer, Integration Service Layer, Application Service Layer, and some vertical layers, as Security, Management and Interoperability services, as illustrated in Figure 1.

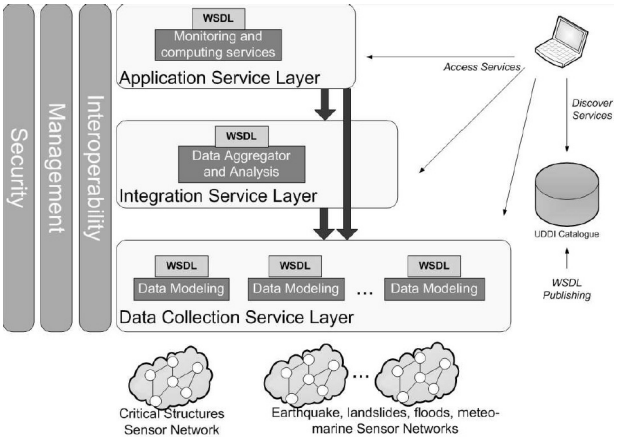

Figure 1. A reference architecture of sensor web services

1. The **Data Collection Service Layer** maps the raw sensed data onto physical reality, and provides to the upper layer an homogeneous view of the networks, a model of the sensors, the measurement and the phenomena is required to complement the knowledge. The Data Collection Service Layer provides the services to access heterogeneous sensor network middleware. Indeed, the services offered in this layer translate proprietary data format in a common data model at higher levels.

2. The **Integration Service Layer** contains services that are not associated with one specific sensor network but rather are global and capture integration and interactions across collections of networks. The services of this layer provide aggregated data sets for the application level. They clusterize network data sets according to the data model defined. Consider for example the case in which different technologies are adopted to sense the same data; while the collection layer is responsible to retrieve such data and represent them, this layer is capable of integrating data related to the same phenomena but observed by different heterogeneous networks.

3. The **Application Service Layer** implement different kind of applications to monitor and elaborate complex data structures from heterogeneous sensor networks. Applications are built by invoking and composing services defined in other layers; they all elaborate complex data structures formatted according to the data model defined and accessed via standard WSDL. Furthermore, each service has its own security policy and it is published in a public registry.

Each layer is able to communicate with other layers through standard WSDL interfaces.

In order to design an open system and manage heterogeneous sensor data sources, in this paper, we propose to improve the SeNsIM platform towards a Service Oriented Architecture based on web services (WS) technologies.

In the following section, we first describe the architectural model of the SeNsIM system and its two main components: the wrapper and the mediator. We then illustrate the SeNsIM-Web architecture which extends the first one by adopting the web services technology and so making the SeNsIM system an available, re-usable and invocable integration service in the World Wide Web.

4 From SeNsIM to SeNsIM-Web

The SeNsIM system is an integration platform for heterogeneous sensor systems. It is not just a middleware for sensor networks, but a more general software architecture which makes possible the deployment of applications based on multiple sensor systems/networks and allows a generic user or an application to easily access data sensed by a network.

SeNsIM provides a unique interface for local networks that allow a generic user to express queries by means of an intuitive query visual language. It was conceived with the aim to bridge the gap between heterogeneous sensor systems and to provide a generic user/application with a unique way to manage, query and interact with them. Nowadays there is a tremendous heterogeneity in the logic for interfacing and collecting data from sensor systems. In most of them an application can directly access to sensor hardware by means of opportune drivers. In many different scenarios, as in WSN, an operating system between the hardware platform and the application allows an easier use of the available sensing functions and often a further middleware layer between the operating system and the user/application provides an easy way to access sensor data. The major issue of an integrating system lies in the heterogeneity of the hardware to sense data and of the repositories, these make data management and retrieval process a hard task to achieve.

Furthermore, sensor data may be differently structured according to the specific representation of different sensor systems. In order to face such problems SeNsIM defines:

- an *architectural model* able to support in an efficient way the management of data even when sensed by different networks;
- a *data model* capable of representing in a unique format both sensor data and sensor systems.

SeNsIM architectural model has been designed by exploiting the *wrapper-mediator* paradigm, a well-known technique to integrate data from heterogeneous source [6], in which the *mediator* component accesses different data sources by means of ad hoc connectors (*wrappers*). In SeNsIM, each wrapper explores and monitors the local sensor network and sends to the mediator an appropriate description of the related information according to the common data model. On the other hand the mediator organizes such information and keeps a unique view of all systems in order to satisfy user or application queries. SeNsIM architectural model is made of 4 logical layers (see figure 2):

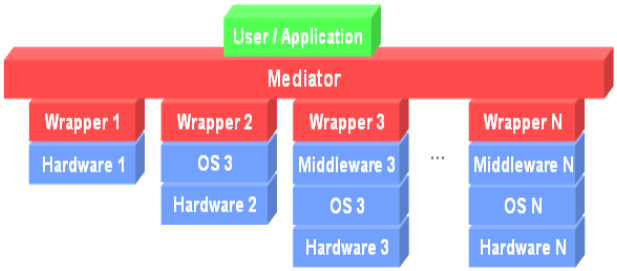

Figure 2. SeNsIM architectural model

1. The **Application** or **User Layer** allows a generic user to submit queries and elaborate the retrieved data; a generic application can also access sensor data through system API. The system provides support for monitoring queries which return the corresponding responses in real-time as well as for event queries. Many context-aware applications need to trigger adequate actions/-countermeasures after that some events have been generated from sensor systems.
2. The **Mediator Layer** aims to classify networks features as well as to format and forward queries to specific wrappers; a DBMS is used to store data related to networks with their sensors, user queries and related results.
3. The **Wrapper Layer** aims to extract and manage information about the underlying network and its sensors; at this layer queries from mediator are received and executed on the local system by using its API and the local query language. A DBMS in each wrapper component is kept to for storing network/sensors information according to the data model and local queries with related results.
4. **The Sensor System Layer** interacts with the wrapper component in order to extract network features and carry out the retrieval process.

SeNsIM data model is able to represent a *sensor node* as well as the whole *network*. According to the model a sensor node is an object characterized by a tuple of information which combine both structural and behavioral features. A network object has to include global information such as type of sensor system, middleware (if present), supported sensor board as well as information related to sensor components (list of sensors, possible list of clusters, topology matrix). Moreover it is possible to specify network global predicates (e.g. average temperature of the network). XML has been used to represent the data model, since it provides platform independence, interoperability and can be easily parsed [7]. XML-based descriptors let have a unifying grammar by which systems can describe their abilities and define a standard language protocol with which the different entities in the framework can communicate. In SeNsIM XML descriptors are exchanged between mediator and wrappers components.

In order to move towards an open and standard architecture, we have improved SeNsIM with a WS interface by implementing a complex *Integration Service* that offers collection and integration functionalities. In such a way the SeNsIM system become a service with a standard interface, which can be invoked on the web through the WS paradigm. Figure 3 shows the SeNsIM-Web general architecture.

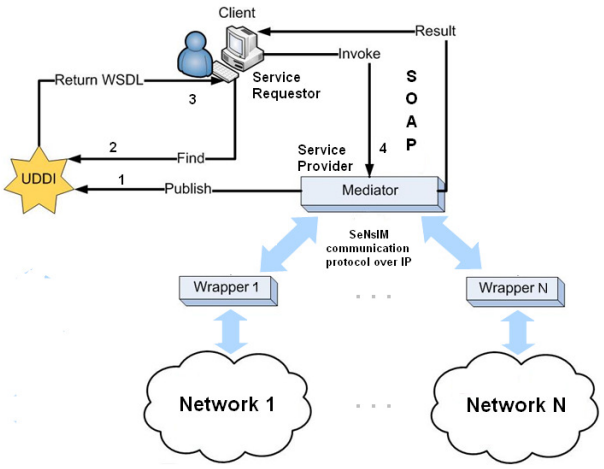

Figure 3. SeNsIM-Web architecture

Mediator acts as *Service Provider* and publishes its services in the UDDI registry in such a way they can be discovered and invoked by a generic user. Third-party applications can discover services as well and invoke them according to their WSDL interfaces. Let us consider a simple retrieval process carried out by a generic user. After receiving the WSDL of the service, he invokes a specific *Monitoring* service (more details in Section 5), through a client application (*Service Requestor*), which allows him to submit a query to a specific network/system. The query is then translated into a SOAP message and sent to the mediator. The latter extracts the query parameters, builds a XML query message and sends it to the apposite wrapper.

The wrapper, first builds the local query according to the underlying system and its query language, then runs it by using available system or middleware API and finally retrieves the results to send (in a XML format) to the mediator. The mediator encodes the data according the defined data model and encapsule them in a SOAP response towards the user application that shows the results to the user.

At the moment, the communication between wrappers and mediator is carried out by using TCP and UDP sockets according to the SeNsIM communication protocol. In the next section we will illustrate some design and implementation details of the whole architecture. In future works we will provide the wrapper with WS interface in order to offer ”data collection services”, too.

5 Design and implementation details

In this section we first illustrate the design principles of SeNsIM-Web by analyzing its main functionalities; then we give some details about the implementation and finally we describe a simple usage scenario of the whole system on a real test bed in order to demonstrate the validity of our approach.

SeNsIM-Web main functions are reported in the following. They have been inherited from the SeNsIM platform and re-implemented as web methods, that are methods exposed as Web Services.

1. **Getting networks** gives as output the list of all networks connected to the system, their topology and basic information such as network identifiers, number of sensors and maximum depth for each network.

2. **Getting network features** gets the network identifier as input parameter and gives in output: type of system (description),the middleware (if present), number of sensors, maximum depth. Further, it gives information about the wrapper which is handling the network: its IP address, registration time and communication ports. Finally it gives information about the base station its working frequency and how it is linked to the wrapper.

3. **Getting sensor details** gets the network identifier as input parameters and gives information of each sensor of the related network: identifier, membership cluster (if defined), type, depth in terms of number of hops to the base station and parent id.

4. **Getting sensor parameters** gets as input a specific sensor identification given by the triple: network identifier, cluster (if defined) identifier and sensor identifier. It gives as output the intrinsic parameters of the sensor, such as the free ram, the voltage and the channel quality.

5. **Getting sensor predicates** gets as input a specific sensor identification and gives as output the sensor predicates, that are all the physical variable a sensor can measure.

6. **Monitoring** carries out a simple monitoring task by querying a sensor or a whole network. It gets as input the query parameters: sensor identification, sample period, duration, the retrieval interval, already defined in SeNsIM as the time interval in which wrappers may collect query results before sending them to the mediator. Moreover, a *where clause* parameter which defines a specific condition of data retrieval (e.g. temperature $> 50°C$). Hence, the query process starts according to the SeNsIM protocol and mediator database can be populated with query results.

7. **Results retrieval** carries out the task of retrieving query results from the mediator database. It gets as input the query identifier, the network identifier or a specific sensor identification (in case of single sensor

related query) and gives as output the query results for each sensor and predicate.

Functions from 1 to 6 are carried out by a user/application, while the last one is called periodically during a monitoring task. Due to space limitations, we do not show the graphic of dependencies between web methods. However is quite obvious that some methods can only be invoked after other ones. For example a generic user can not get a specific network features (second function/method) without having discovered all the networks connected to the system with their identifiers (first function/method). According to that, we ordered the above list of functions like a simple usage scenario to carry out a retrieval process. For each method has been defined a WSDL interface describing how to access the service to invoke its operations. The interfaces are then published in the UDDI registry to be discovered by a generic service consumer.

We used Apache Tomcat [16] as a server to deploy our services and *Axis* (Apache eXtensible Interaction System) as SOAP Engine. Axis [19] is a framework which include a java class generator from WSDL files. The main classes generated in this process (called *stub* on client side and *skeleton* on server side) realize the binding between client (which invokes the service) and server (where the service is) by creating SOAP messages and encapsulating them in HTTP messages. Another class generated on server side contains the structure of the service which should be filled with the service implementation code.

We set up a first experimental testbed with two TinyDB based networks integrated by the SeNsIM system. Each network is composed of Crossbow MICA sensors running the TinyOS [21] operating system. TinyDB [20] is a middleware for WSN which provides a query processing system for extracting information from a network of TinyOS sensors. It is just a simple way to verify the validity of our approach, however in the future we are planning to continue the experiments in order to evaluate system scalability and performances.

Figure 4 shows the main screen of the client graphical interface and the details of the network 1 after having called the related method through the button *Network Details* on the left functions panel.

Figure 5 shows the query panel of the monitoring function. After the insertion query parameters, user can call the service by clicking the *Send* button. We chose to retrieve all the predicates of the two sensors of network 1. Query results are shown in a panel which has a table for each sensor in the network, reporting the values of predicates (see figure 6). Tables are updated in real time with new results. In the same figure the graphical visualization of the results related to a specific sensor (sensor 1) is shown so that user can monitor in an easier way the trend over time of the requested predicates.

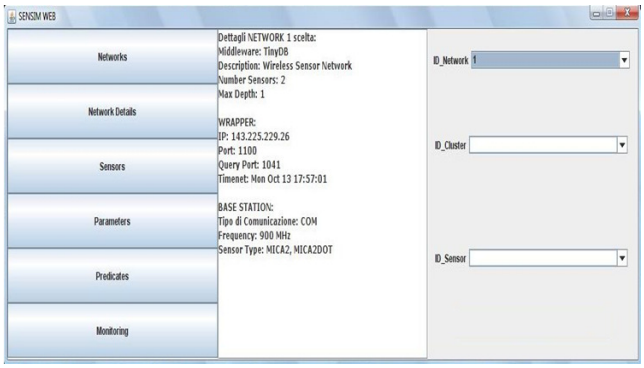

Figure 4. SeNsIM-Web main screen and network details function output

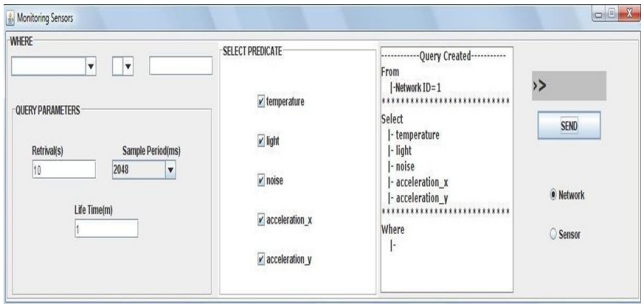

Figure 5. Query panel of the monitoring service

6 Conclusion and Future Works

In this paper we have presented SeNsIM-Web, a web services-based integration platform for sensor networks. It allows a single unified view of a lot of systems and enables a flexible deployment and interconnection between them, even if located in different places.

The core architecture is based on the Mediator/Wrapper paradigm in order to provide a layered and scalable architecture, furthermore, the Mediator has been provided with a Web Service Interface, in order to let its functions and data be accessible via standard protocols. In order to give a more layered view of the whole system, we are going to provide the wrapper with a standard WS interface (to develop autonomous collection services), too. Furthermore we will explore new standards and potentiality of the semantic web to improve the adopted data model.

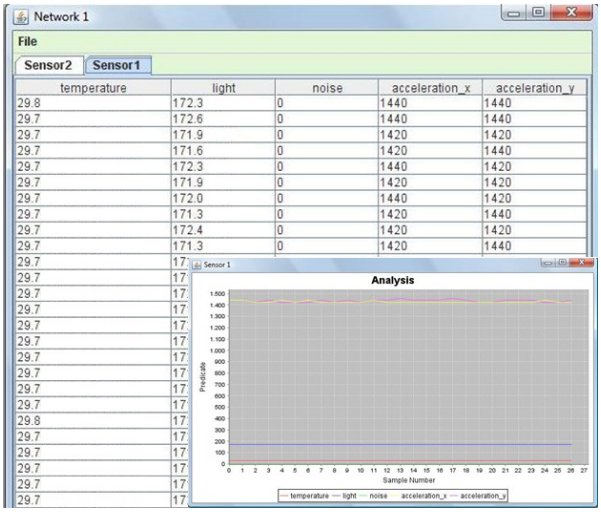

Figure 6. Results windows

References

[1] SensorML - URL: http://www.opengeospatial.org/standards/sensorml.

[2] O&M - URL: http://www.opengeospatial.org/standards/om.

[3] V. Casola, A. Gaglione, A. Mazzeo, "A Reference Architecture for Sensor Networks Integration and Management", in Proceedings of the 3rd International Conference on Geosensor Networks (GSN 2009), Oxford, UK, July 2009. To appear.

[4] F. Flammini, A. Gaglione, N. Mazzocca, V. Moscato, C. Pragliola, "Wireless Sensor Data Fusion for Critical Infrastructure Security", in Proceedings of the 1st International Workshop on Computational Intelligence in Security for Information Systems (CISIS'08) - Advances in Soft Computing, pp. 92-99, 2008.

[5] F. Flammini, A. Gaglione, N. Mazzocca, C. Pragliola, "DETECT: a novel framework for the detection of attacks to critical infrastructures", in Proceedings of the European Safety and Reliability Conference (ESREL'08), Valencia, Spain, Sep. 2008.

[6] G. Wiederhold, "Mediators in the architecture of future information systems", IEEE Computer, 25(3), 1992.

[7] W3C Architecture Domain, Extensible Markup Language (XML). URL: http://www.w3.org/XML/.

[8] X. Chu and R. Buyya, "Service Oriented Sensor Web", Sensor Networks and Configuration (2007), pp. 51-74.

[9] A. Sheth and C. Henson, S. Sahoo, "Semantic Sensor Web", IEEE Internet Computing, vol. 12, no. 4, 2008, pp. 78-83.

[10] C. Alegre, C. Monfort, P. Lazaridis, "WIN : a new Service Oriented Architecture for risk management", www.vfdb.de/riskcon/paper/Interschutz Conference -WIN Paper.pdf

[11] K. Aberer, M. Hauswirth, A. Salehi, "The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks", Technical Report, 2006.

[12] S. Ahn, K. Chong, "Building a Bridge for Heterogeneous Sensor Networks", Proceedings of the Fourth IEEE Workshop on SEUS-WCCIA 06.

[13] T. Kobialka, R. Buyya, C. Leckie, R. Kotagiri, "A Sensor Web Middleware with Stateful Services for Heterogeneous Sensor Networks Intelligent Sensors", Sensor Networks and Information, 2007, ISSNIP 2007.

[14] http://sensorweb.geomatics.ucalgary.ca/.

[15] S. Liang, A. Coritoru, C. Tao, "A Distributed Geo-Spatial Infrastructure for Smart Sensor Webs", Journal of Computers and Geosciences Vol.31, 2005.

[16] http://jakarta.apache.org/.

[17] WIN Project - URL: http://www.ist-world.org/ProjectDetails.aspx?ProjectId=79df0ee297c34606b55ad87a6f3c247d.

[18] ORCHESTRA Project - URL: http://www.eu-orchestra.org/overview.shtml

[19] http://ws.apache.org/axis/java/architecture-guide.html.

[20] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An acquisitional query processing system for sensor networks", ACM Transactions on Database Systems, 30(1):122-173, 2005.

[21] TinyOS Project. URL: http://www.tinyos.net.