The 6th International Conference on Intelligent Environments - IE'10
Kuala Lumpur, Malayasia July 19-21, 2010, pp. 209 - 214.

Sensors as a Service Oriented Architecture: Middleware for Sensor Networks

John Ibbotson, Christopher Gibson, Joel Wright, Peter Waggett, IBM U.K Ltd, Petros Zerfos, IBM Research, Boleslaw K. Szymanski, Rensselaer Polytechnic Institute, David J. Thornley, Imperial College London
{john_ibbotson, gibsoncr, joel.wright, peter_waggett }@uk.ibm.com, pzerfos@us.ibm.com, szymab@rpi.edu, djt@doc.ic.ac.uk

***Abstract*: There is a significant challenge in designing, optimizing, deploying and managing complex sensor networks over heterogeneous communications infrastructures. The ITA Sensor Fabric addresses these challenges in the areas of sensor identification and discovery, sensor access and control, and sensor data consumability, by extending the message bus model commonly found in commercial IT infrastructures out to the edge of the network. In this paper we take the message bus model further into a semantically rich, model-based design and analysis approach that considers the sensor network and its contained services as a Service Oriented Architecture. We present an application of a hierarchic schema for nested service definitions together with an initial ontology that describes the assets and services deployed in a sensor network infrastructure.**

***Index Terms*: Sensor networks, Ontology, Service composition, Service modeling**

I. Introduction

The diversity of sensors, actuators and networking technologies used in intelligent environments provides significant challenges in the areas of identification and discovery, access and control, data consumability and trusted policy-based interoperability. The ITA Sensor Fabric [1,3], developed as part of the International Technology Alliance in Network and Information Science [2], has addressed these challenges to provide an extensible middleware layer to interconnect sensors with users (human or software agents) that need to consume the data generated by them. The Sensor Fabric (or Fabric) extends the message bus architectural model to the edge of the network. It spans between the reliable communications infrastructures found in data centers and the intermittent connectivity of deployed sensors and mobile personnel connected using ad hoc wireless network technology. The Fabric provides universal access to sensor data from any point on the network. It maximizes the availability and utility of the data to users, whilst hiding the complexity of the underlying network infrastructure.

Research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

The Fabric is an extensible middleware platform with each participating node interconnecting with neighbours to form a lightweight service bus. Its plug-in architecture allows new functions such as filters, transformations, policy enforcement, security, data fusion and event detection algorithms to be easily deployed within the network and selectively applied to sensor messages as they flow through the network.

In addition to its use with deployed sensors, the Fabric is also used as a research and development tool. New algorithms can be tested using the Fabric's record and playback, sensor simulation, and performance measurement support. New sensors can trialed in an environment that bridges between simulation and fielded systems.

Central to the functioning and management of the message bus is a distributed Registry, an evolution of the Service Registry commonly used with a Service Oriented Architecture (SOA). Built using the ITA Gaian dynamic distributed federated database technology [4], the Fabric Registry is used to track all aspects of the message bus' operation including assets, users, topology, and plug-in functions. The Fabric Registry also tracks tasks being performed using the Fabric. These are groupings of sensors and users that are assigned to some activity, for example the users and assets associated with water level monitoring in a flood detection system. Tasks may be used to prioritise resources interconnected using the Fabric; for example, data from one geographical area may be given priority over another in the case of extreme weather conditions leading to flooding. The Fabric does not establish task priorities itself; this is left to external applications that consume the data provided through the Fabric.

The Fabric has been developed as a lightweight service bus for sensors, which is intended to augment existing Enterprise Service Bus technologies. Sensing environments provide different challenges to those in highly reliable business infrastructures and the Fabric has been designed with these in mind. However, there are advantages in thinking of the sensor assets and the plug-in algorithms deployed onto the bus as services in the context of a Service Oriented Architecture (SOA). This has previously been discussed in [5], but this paper focuses on extending the *message bus at the edge* model to a *service bus at the*

edge. We describe a semantically rich, model-driven approach to a practical SOA for sensor networks.

In the remainder of this paper, we first present a motivating use case for an SOA on a sensor network middleware in section 2. In section 3 we describe an approach to modeling services on the sensor network using annotated UML Activity diagrams, which is expanded to include service composition in section 4. In section 5 we describe a draft ontology for modeling the assets and services contained within the network. Finally we end with some conclusions and outline further work to develop an environment for describing, modeling, analysing and deploying services onto the sensor network SOA infrastructure.

II. A Motivating Scenario

There is a trend in healthcare that encourages the long-term treatment of patients in their own homes for both clinical and economic reasons, particularly for elderly patients who value their independence. Sensor technology can aid this by monitoring both the patient's vital signs (blood pressure, temperature, pulse rate etc.) and activities (movements, utterances, noise) using mobile sensors attached to the patient or fixed sensors located around the home. In addition, sensors can monitor a patient's living environment and daily activity patterns. For example, smart software agents may use motion sensors within the home to learn about the patient's normal behavior patterns and detect any deviation from that behavior; if there is no movement within the home by 8:00am, then issue an alert. This could be augmented by personal alarm systems activated by the patient in case of distress. In the same way that home automation and personal computers are linked by wireless technology, similar networking techniques may be used to create a home medical hub node *(MediHub)* that integrates home sensors for medical monitoring. These sensors may communicate with the hub by different network protocols such as Zigbee [6] or Bluetooth [7]. Using a message bus model, such as that provide by the Fabric, this infrastructure can be connected with emergency services and other relevant patient support to provide effective patient care.

We illustrate this with a use case within this scenario that shows the response to an emergency patient alert generated by the *MediHub* node within a patient's home. Note that the emergency response described in this use case is a specific task within the context of a longer running patient healthcare management process.

In response to the detection of an emergency in the patient's home the *MediHub* node publishes an alert *(MedAlert)*, which is received by the ambulance dispatch service (which will have subscribed to all alerts from all *MediHub* nodes in its locality) and the local hospital (which will have subscribed to alerts from all patients under its care). The alert triggers the dispatch of an ambulance to the patient's address. Whilst the ambulance is in transit, paramedics in the ambulance and medical teams in the hospital's accident and emergency (A&E) department automatically receive the patient's sensor medical data *(MedData)*. This allows the hospital team to provide predictive diagnosis to the paramedics before their arrival at the patient's home.

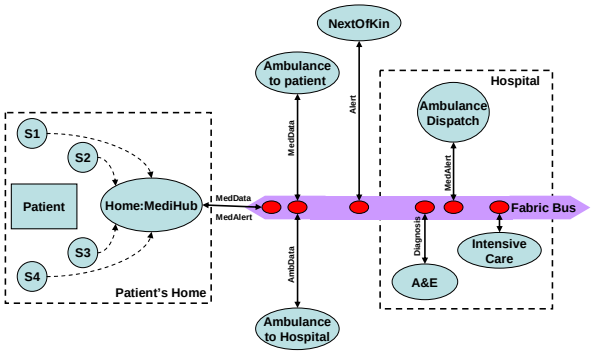

Fig 1: A Medical use case for the Sensor Fabric

Once the patient has been transferred to the ambulance, the patient vital signs monitored within the ambulance provide data feeds via the Ambulance node *AmbData* to the hospital that in turn can continue to provide diagnostic information. The patient monitoring data is also used to prepare other resources in the hospital, such as intensive care, for the reception of the patient. The medical data transferred via the sensor network message bus will include other data such as GPS location from the ambulance so that the dispatcher and medical teams can monitor its progress en route to the hospital. Policy-driven transformations can transcode the data to different formats and resolutions that are appropriate for display and consumption by various devices (e.g. smart-phones, screens on board ambulances, hospital medical record systems, etc.) with heterogeneous display and processing capabilities.

In addition to the medical staff, members of the patient's family *(NextOfKin)* can subscribe to *MedAlert* events and information about the patient (such as the name of the hospital to which the patient is taken) using, for example, a Smartphone application or a message bus/SMS bridge. Policies for access control and encryption within the message bus ensure that privacy regulations are met as messages flow across the bus between the patient's *MediHub*, the ambulance dispatch service, the hospital, the ambulance and the patient's family members. Such policies are applied at organizational, user role, and individual user identity levels.

III. Services on the Sensor Network

Services on the sensor network do not follow the conventional view of a service requestor invoking a service provider that responds to the invocation. This request/response service model is widely used in Web service-oriented SOA applications but is not appropriate for the stream-based services found in an event based messaging environment such as a sensor network. Whilst the Fabric does provide a message bus that is comparable to

those used in mainstream SOA architectures, interconnecting the sensor network's assets and users, the service composition model that is required to *efficiently* deliver composite services built from these individual functions is very different to the process choreography [8] that is commonly used. In constrained stream-oriented environments such as this it is necessary to revert to first principles and discuss how a user could describe a set of services that provides a particular processing function.

For modeling services on the sensor network, we use the UML activity diagram as a starting point. Activities match the semantics of sensors and services in the network, with them being synonymous with services having zero or more inputs and zero or more outputs. This is illustrated in Fig 2, which shows the UML model of part of the sensor network deployed within a patient's home.

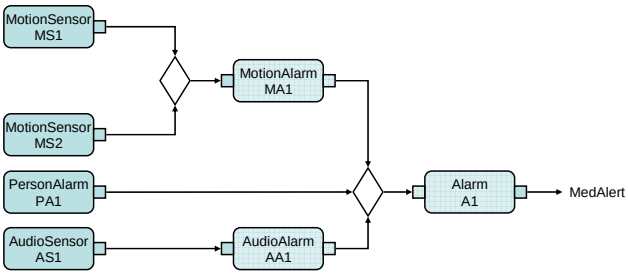

Fig 2: Sensors and services generating an alarm within a patient's home

In this model, there are 4 sensors deployed in the home to monitor the environment. Two motion sensors (*MS1* and *MS2*), positioned so as to detect the patient's normal activities (such as getting up from bed), feed a *MotionAlarm* activity *MA1* which generates an alarm event if a variance from normal activity is detected. The *PersonAlarm* sensor *PA1* is carried by the patient and activated in case of emergency. Finally, an audio sensor AS1 can trigger an alarm via an *AudioAlarm* activity *AA1*. Any of these three alarms can cause a *MedAlert* alarm to be published by the *Alarm* activity *A1* from the patient's home onto the Fabric bus which will trigger a response by the hospital emergency services. The UML merge activity may be implemented either as a separate service or as a function of the input port to the *Alarm* service.

The example illustrates several components of the core model: activities, inputs, outputs and connections. UML allows each of these design model components to be annotated with additional information and we propose the exclusive use of semantic annotations to enrich the model.

The UML semantic annotations are made according to a flexible, extensible ontology. This combination (i.e. UML model plus semantic annotation) provides a flexible basis for both transformation and analysis of the model.

Annotations may reflect both abstract and concrete properties of the design required for transformation into a variety of formal models for verifying quantitative and qualitative properties of a model. These opportunities include evaluation of the performance characteristics of services deployed on a sensor network by translating them into, for example, Performance Evaluation Process Algebra (PEPA) for integration into the MARS framework that explains information quality and effectiveness and will serve to quantify the related medical demands on time and resource consumption in a manner that supports decision making [9]. Further annotations will allow transformation of the model into other representations including transformation of the model to generate a deployment descriptor describing the deployment of an implementation of the model on the Fabric, or indeed code generation where the model describes new or incomplete components.

We expect transformations into formal models to be used as part of an initial design process prior to deployment in the sensor network. Results from the analysis of the formal model may then be used to enhance or modify the annotations on the design model. The flexibility of semantically annotating the model gives us a technique to easily integrate the results back into the original UML model as feedback to the designer/developer. We describe this as *round-tripping* (Fig 3).

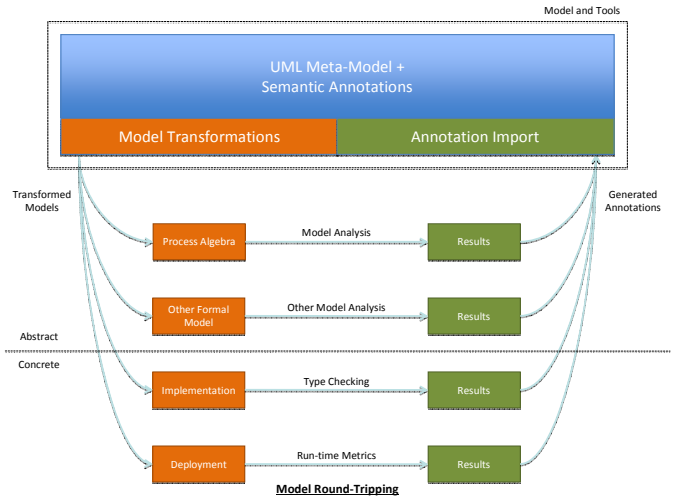

Fig 3: Round-Tripping from the Core Design Model

Semantic annotations can also be used to capture physical properties of a deployed sensor network. This supports the second, concrete, round-tripping route shown in Fig 3 between the design model and the services deployed on an active sensor network. The Fabric provides instrumentation on the sensor network message bus allowing real-time metrics to be gathered and a profile of the performance of the deployed services to be generated. This profile can be used to provide further annotations within the design model to allow subsequent refinements of the formal representation that will be more suitable for the deployment environment. We expect the UML design model to undergo multiple round-trips between the formal and deployed states during its lifecycle.

IV. Service Composition

Services may themselves be composed of other services which results in a requirement for a hierarchic composition schema that describes their interconnection. Hierarchic interconnection of components is a generic requirement in silicon CAD tools, which has led to a schema known as the *five-box schema*. A description of the schema can be found in the Silicon Integration Initiative physical design language specification [10].

The five-box schema relates the definition of a component and the ports (input and output interfaces) it exposes to instances of the component. The concept of a component in the silicon CAD domain relates exactly to a service in the sensor network domain. The schema, labeled to describe sensor network services, is shown in the following figure.

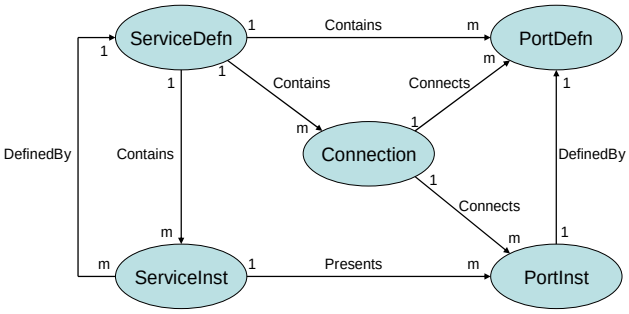

Fig 4: The Five-Box Schema for Hierarchic Services

In the schema, a service definition *(ServiceDefn*) contains multiple port definitions *(PortDefn*), which may be specialised as either an input or output. Service definitions may also contain instances of services *(ServiceInst*) which creates the hierarchic service description. Service definitions also contain the connections (*Connection*) that interconnect the service port definitions. Services and ports can be instantiated (*PortInst*) and interconnected with the instances linked by the *DefinedBy* relationship to their definitions. We have used the five schema elements, service and port definitions and instances together with connections, to provide the anchor points for semantic annotations supporting the UML activity model for sensor network service designs. These classes map directly to the activities, typed ports and connections of the UML as shown in Fig 2.

Applying this schema to the example in Fig 2, the service *MotionSensor* is used twice as instances *MS1* and *MS2*. Such a dual use of *MotionSensor* would be represented in the schema as a single instance of the *ServiceDefn* class for the *MotionSensor* and two instances of the *ServiceInst* class; one for *MS1* and one for *MS2*. The port definition for the service will be annotated with the type of the message generated by the *MotionSensor* service. Similarly, service definitions will be provided for the *PersonAlarm*, *AudioSensor*, *MotionAlarm*, *AudioAlarm* and *Alarm* sensors. The instances of these services *PA1*, *AS1*, *MA1*, *AA1* and *A1* and their associated ports will be included in the model. Connections between the instantiated ports also form part of the model. Type enforcement will ensure that the connection matches the types of the output and input ports it is interconnecting. Additional annotations on a connection (such as its latency or rate) are included as required by the target model transformations (for example a PEPA transformation).

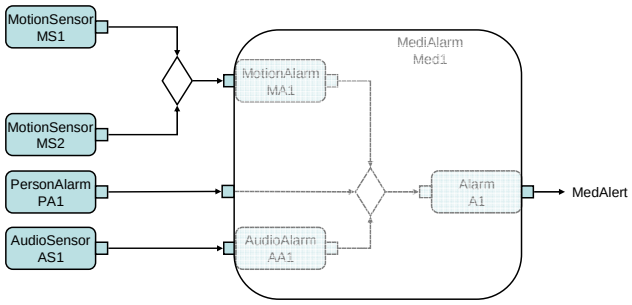

Fig 5: Composing Services

Services may form part of other services as illustrated in Fig 5. In this example, the *MotionAlarm*, *AudioAlarm* and *Alarm* services together with the output of the *PersonAlarm* have been identified as a useful, reusable service. The five-box schema allows this collection of interconnected services to become a service definition (*MediAlarm*), which can be added to a design tool's palette of available services for future use. The instantiation of this composite service as *Med1* in a design tool would allow users to expand the composite service to inspect the atomic or other composite services that make up the *MediAlarm* service description. The five-box schema supports the hierarchic definition of composed services to any level but in practice, the depth of the hierarchy will be limited.

In addition to the benefits of composition in a service design environment, providing a rich annotation capability for the core model creates further opportunities for *automating* the composition of services. This may not necessarily involve the hierarchical composition described earlier, but automated composition is important in environments that must be adaptive and self-managing; this is particularly true for the stream-based services in a sensor network.

We have identified four possible scenarios where the automated composition of services is advantageous:

1. **Information typing.** Users may require information of a type that is available from a particular service output. Automated composition will allow the tree of services to be constructed to provide the correctly typed information for a user.
2. **Service optimisation.** When transforming the core model into a deployment model, there may be multiple compositions of services that meet the same functional requirements. Round-tripping of the core model through formal or deployed models may identify which

of the alternative compositions are optimal (based on network resource utilization or some other metric) for a given network configuration.

3. **Functional redundancy and substitution.** In sensor networks where there is an unreliable network and system layer, services may become inaccessible due to communication breakdown or energy depletion of sensor nodes. Functional resilience may be achieved by re-composing services within the remaining accessible network to provide the same functionality. In case no exact equivalent functionality can be provided through re-composition due to unavailability of appropriate service instances, a substitute service that implements only a subset of the required functions can be alternatively suggested. This assumes a sensor network environment that is regularly monitored and in which services can be re-composed and deployed autonomically.
4. **Run-time optimization**. During a services execution the network conditions can drastically change, for example the deployment of new services on to a shared node could cause it to become overloaded. Hence, periodic load balancing and/or redistribution of long term running services may improve their performance. This is particularly important for composite services which can allocate their component services to nodes in such a way as to balance both the communication and computational load in the network.

V. An Ontology for SOA on a Sensor Network

Various ontologies and vocabularies have been proposed for sensor networks in the literature with SensorML [11] and OntoSensor [12] being probably the best known. Others within the ITA research programme have addressed semantic techniques in the allocation of resources in sensor networks [13, 14]. The ontology presented in this paper is not intended to replace those presented elsewhere. Instead, it is used to describe the core sensor network design model, the hierarchic five-box schema for services and the artifacts represented in the existing Fabric registry relational database schema. Its purpose is to provide a flexible, extensible representation of the core model and the annotations needed for transformation of the core model for analysis, implementation, and deployment.

Fig 6 shows the initial set of classes within the ontology. For clarity, the relationships between the classes are not shown. Each Owl file describes the classes within a different namespace. The *SemanticFabric* namespace acts as a top level entry point whose role is simply to import other namespaces into the ontology. Instance ontologies such as one to describe the earlier medical example will then import the SemanticFabric namespace to access definitions of the available classes.

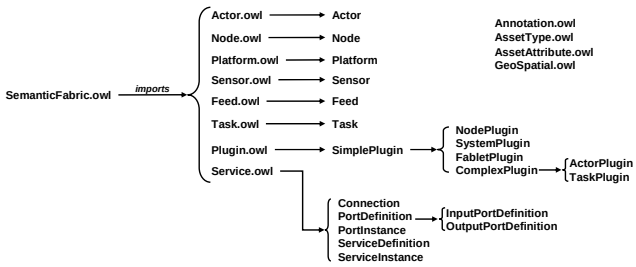

Fig 6: A draft ontology for sensor networks

There are four major groups of classes in the ontology: Assets, Tasks, Plugins, and Services. Assets include Nodes, Platforms, Sensors and their data feeds (that generate the event streams subscribed to by sensor network users). The users are represented in the ontology as Actors and are either end (i.e. human) users or software processes. Assets can have Types and Annotations associated with them in the form of their affiliation, the asset's readiness, their roles, security credentials and availability. Assets can also have geographical information associated with them via the GeoSpatial class which may include information such as the asset's latitude, longitude and altitude. Mobile assets may have information on their velocity and current bearing.

A set of classes support the Fabric notion of Tasks. These are groupings of sensor network assets that have been allocated to a particular activity. The concept of a Task is provided to enable external planning and resource applications to track and prioritize the allocation of assets within the Fabric.

Classes are also provided within the Service namespace to support the five-box schema representation of hierarchic services with the *PortDefinition* class having sub-classes to specialise input and output ports for a service.

Fig 7 expands the classes for representing hierarchic services to show the OWL predicates that link the classes. In the ontology, *Connections* have a "from" and "to" qualification linking them to *PortDefinition* and *PortInstance* classes to represent the source and sink port classes they interconnect.

Finally, there are a set of classes to support the Fabric plug-in architecture. When deployed, services will be implemented as plug-in modules that process messages as they flow through Fabric nodes. *Node plug-ins* process all messages that flow through a node. *Actor* and *Task plug-ins* are applied to messages destined to a particular actor or flowing as part of a defined Task. *Fablets* have the additional flexibility of being able to interact with other non-Fabric applications and resources. Finally, Service plug-ins extend the capability of the core Fabric functionality.

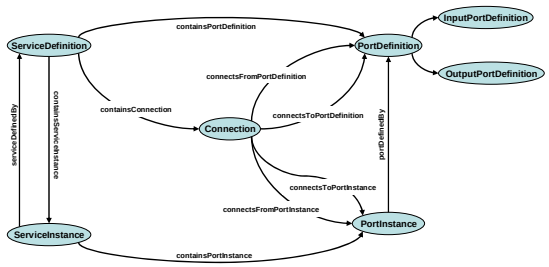

Fig 7: The Five Box Schema represented as an Ontology

VI. Conclusions and Further Work

In this paper we have described a model-based approach for the design, analysis and deployment of services within a sensor network based on previously reported work on the development of sensor network middleware: the ITA Sensor Fabric. Using a motivating example scenario, we base our core design model on UML Activity diagrams and their associated semantics, with an underlying representation using the five-box schema realized as an ontology. The model may be transformed into alternative models (including formal models such as process algebras) for analysis with the results providing additional annotations to the core design model via a round-tripping mechanism. The core model may also be transformed into a form that can be deployed onto a real sensor network using the ITA Sensor Fabric.

Further work is continuing under the ITA research programme to develop a tooling infrastructure to support the creation, transformation, analysis and deployment of sensor network service models. User tools based on the Eclipse framework are being developed which will integrate with a semantic wiki provides an interactive representation of the knowledge it contains. This will provide a flexible environment where different applications, each using the core sensor network model, can be developed. Other applications that are based on semantic technologies such as mission and resource planning will benefit from the flexibility of the tools framework which will permit application specific ontologies to be imported in the form of RDF files and integrated with the core sensor network model.

A version of the ITA Sensor Fabric can be downloaded for evaluation and experimentation from the IBM Alphaworks website at http://www.alphaworks.ibm.com/tech/fabric4sensors.

References

[1] J. Wright, C Gibson, F. Bergamaschi, K. Marcus, R. Pressley, G. Verma, G Whipps, “A Dynamic Infrastructure for Interconnecting Disparate ISR/ISTAR Assets (The ITA Sensor Fabric),” *IEEE/ISIF Fusion 2009 Conference*, July 2009.

[2] G. Cirincione and J. Gowens, “The International Technology Alliance in Network and Information Science a U.S.-U.K. Collaborative Venture,” *IEEE Comms. Mag*., vol 45, pp 14-18, March 2007.

[3] Fabric for Sensor Network Management and Data Transfer, http://www.alphaworks.ibm.com/tech/fabric4sensors

[4] G. Bent, P. Dantressangle, A. M. D. Vyvyan and V. Mitsou, “A Dynamic Distributed Federated Database,” in *Second Annual Conference of ITA*, September 2008.

[5] J. Ibbotson, S. Chapman, B. K. Szymanski, “The Case for an Agile SOA”, *Annual Conference of ITA, 2007*, September, 2007.

[6] The Zigbee Alliance, http://www.zigbee.org/Home/tabid/188/Default.aspx

[7] The Bluetooth Special Interest Group, http://www.bluetooth.com/bluetooth/

[8] Web Services Business Process Execution Language Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

[9] D. Thornley, R. Young, J. Richardson, “Development of a Mission Abstraction Requirements Structure (MARS) and Stochastic Modelling for Sensing Service-Driven Mission Performance Prediction”, Imperial College Dept of Comp Tech Report 2009 #10, http://www.doc.ic.ac.uk/research/technicalreports/2009/DTR09-10.pdf

[10] Silicon Integration Initiative, “CHDStd Reference Specification Physical Design Language (PDL) Description”, http://ftp.si2.org/si2_publications/CHDStd/PDF/04_pdl_desc.pdf

[11] A. Robin, S. Havens, S. Cox, J. Ricker, R. Lake and H. Niedzwiadek. “OpenGIS Sensor Model Language (SensorML) Implementation Specification”. Technical Report, Open Geospatial Consortium Inc, 2006.

[12] D.J. Russomanno, C.R. Kothari and O.A Thomas. “Building a sensor ontology: A practical approach leveraging ISO and OGC models”. In Proceedings of the 2005 International Conference on Artificial Intelligence (ICAI), pp 637-643, 2005.

[13] M. Sensoy, T. Le, W. W. Vasconcelos, T. J. Norman, A. D. Preece, “Resource Determination and Allocation in Sensor Networks: A Hybrid Approach”, The Computer Journal, January, 2010.

[14] M Gomez, A Preece, M P Johnson, G de Mel, W Vasconcelos, C Gibson, A Bar-Noy, K Borowiecki, T La Porta, D Pizzocaro, H Rowaihy, G Pearson, & T Pham, An Ontology-Centric Approach to Sensor-Mission Assignment, Proc 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008), in press, 2008.