# Service Oriented Sensor Networks for Mobile Computing

Haeng Kon Kim

***Abstract*— Mobile applications execute in an environment characterized by scarce and dynamically varying resources. We believe that applications have to adapt dynamically and transparently to the amount of resources available at runtime. To achieve this goal, we use the conventional extension of the client server model to a client-proxy-server model. The mobile devices execute the client, which provides the user interface and some part of the application logic. The proxy is a component of the application that executes in the wired network to support the client. As the user moves, the proxy may also move to remain on the communication path from the mobile device to a fixed correspondent host. Logically, the proxy hides the "mobile" client from the server, who thinks it communicates with a standard client (i.e., a client that executes on a powerful desktop directly connected to the wired network). Wireless sensor networks (WSNs) provide various environment data in the real-world, and also WSNs´s middleware is able to offer field data in real-time by user queries. WSNs also play an important role in ubiquitous computing with RFID technologies currently, and have evolved from many studies and being advanced to the future. In this paper, we propose a service-oriented sensor ontology which enables services based on service-oriented properties for materialization of the future ubiquitous computing. In contrast to legacy approaches, this paper defines the new service classes (Services, Location and Physical), as well as their properties and constraints that enable the service-oriented service based on service properties. We also have regard to reuse of ontology, service classes were designed to link with legacy OntoSensor ontology.**

***Index Terms*— Service-Oriented Architecture, Mobile Computing Wireless Sensor Networks, RFID, Mobile Application**

## I. Introduction

The mobile devices execute the client, which provides the user interface and some part of the application logic. The proxy is a component of the application that executes in the wired network to support the client. As the user moves, the proxy may also move to remain on the communication path from the mobile device to a fixed correspondent host. Logically, the proxy hides the "mobile" client from the server, who thinks it communicates with a standard client (i.e., a client that executes on a powerful desktop directly connected to the wired network). Sensor networks are dense wired or wireless networks for collecting and disseminating environmental data. They consist of a large number of sensor nodes that are connected to central processing nodes called gateways. These networks are characterized by three main features.

First, they are highly dense so that hundreds or thousands nodes may be deployed in limited geographical areas. These nodes return huge amount of data that must be efficiently searched to answer user queries. Unfortunately, classical information retrieval techniques showed poor performance in searching sensor networks data as they return many false positives/negatives. Second, many of the captured data are analogous in nature making the chance of finding a specific term quite good. Most sensors are characterized by similar calibration mechanisms that can be described using different terms. String matching search techniques may not retrieve all relevant data because different words/terms were used that did not match directly the term. This compromises the performance of the search engine. A big improvement in search engine performance could be achieved if these relationships are captured and utilized, and this is exactly what ontology can do. This was demonstrated in some recent work on the use of process of ontologies [1, 2] that showed an increase in the precision of service discovery queries when semantic representations were used over syntactic representations.

Third, high filtering must be required when either the user makes full use of this information or the provider offers the user this information because legacy sensor ontology has been designed to manage the sensor networks resources under focus of physical approach. However, first of all, the user will be expected that they much prefer property information (temperature, humidity, pressure, and so on) to physical information of sensor networks in future ubiquitous environments [3, 4], may also need the location property joined with it.

In this paper we propose the service-oriented sensor ontology which enables service-oriented services in future ubiquitous computing. We also have regard to reuse of ontology, Service, Location and Physical classes were designed to link with legacy OntoSensor ontology, and its properties and constraints were also defined newly as service-oriented service. Even if service-oriented service focused on property of the sensing data has differences with legacy ontology, it has compatibility with them under semantic technologies.

## II. Related works

### 2.1 **MOBILE APPLICATIONS**

To define a suitable architecture, we first identify categories of applications a mobile user is most likely to execute on his mobile device. Due to the existing limitations of portable devices (limited computational power, disk space, screen

Haeng Kon Kim is a Professor of School of Information Technology, Catholic University of Deagu, Korea. (e-mail:hangkon@cu.ac.kr )

size, etc.), we claim that portable devices should not be considered general purpose computers. Even though portable devices will become increasingly powerful, they will never match the computational power and facilities available on typical desktop machines. [5, 6]

Similarly, while the wireless technology will improve, providing more and more bandwidth to the end user, wired network technology will advance as well, with the result that wireless networks will remain, in the near to medium future, orders of magnitudes slower. Therefore, mobile computing will always be characterized by a scarcity of resources, relatively speaking. In our opinion, an end-user will execute applications in one of the following six categories in such an environment:

- Standalone applications such as games or utilities
- Personal productivity software (word processors presentation software, calendars)
- Internet applications such as e-mail,WWW browsers, multi-user calendars, or telnet
- Vertically integrated business applications (field installation and services, security)
- New "location-aware" applications: tour planners, interactive guides
- Ad-hoc network and groupware applications.

The first category was originally of little interest to us, since these applications do not involve communication. However, the main idea underlying our architecture is to transparently support resource-constrained mobile devices by powerful proxy servers. We are therefore currently exploring how to generalize this idea to support standalone applications as well. Applications in the second category will be used on multiple platforms: a user will have a version of his/her favorite word processor executing on a laptop as well as on the more powerful desktop in the office. This requires the exchange and synchronization of documents between the machines. Depending on the prevailing view of available network connectivity, two possible approaches are imaginable. Windows CE and MS Office exemplify a first solution. To facilitate access to the Internet, only the client side of the application can be adapted to function well in the dynamic and resource constrained mobile environment. The architecture proposed below is intended for applications in this category. Vertically integrated business applications are often structured as client-server applications. Furthermore, the backends (servers) have to support both existing wired desktops and wireless mobile devices. One example is a bank, where the back office has to support account managers in branch offices as well as mobile customer service representatives.

The clients executing on the portable devices face challenges similar to those faced by traditional Internet clients. They have to adapt to the limitations of the portable device in a dynamically changing execution environment.

To facilitate the deployment of mobile applications, solutions should be transparent to the servers. Due to these similarities, we believe that the architecture proposed below applies equally well to this group of applications.

The location-aware applications exploit the fact that a user is mobile. Possible examples include travel guides, which might display the shortest path from a user's current location to the closest/cheapest/best Italian restaurant, or applications that allow a user to print a document on the closest color postscript laser printer. To the extent that these applications utilize the existing Internet (discovering and accessing nearby resources, for example), the architecture described below can be of value here as well.

Applications in the final category arise out of the mobility of a number of users, for example the meeting of a number of researchers or managers, each equipped with a portable device. Users might want to establish ad-hoc networks to exchange documents (the newest version of the transparencies for the invited talk) or to execute groupware applications to update a shared business plan. Similar to standalone applications, we are however exploring how to generalize our ideas to support ad-hoc network applications.

## 2.2 **MOBILE SENSOR DATA ONTOLOGY**

The term ontology can be defined as "an explicit formal specification of a shared conceptualization". An ontology comprises three components: first, classes or concepts that may have subclasses to represent more specific concepts than in super-classes, second, properties or relationships that describe various features and properties of the concepts, also named slots or roles, third, restrictions on slots(facets) that are superimposed on the defined classes and/or properties to define allowed values (domain and range). Individuals can be defined simply as instances of the classes and properties. The ontology together with a set of instances of classes and slots constitute the knowledge base. Reference [7] presents a detailed description of the development stages of ontologies. Also, many advantages of ontology design are explained in [8], including: sharing common understanding of the structure of information among people or software agents, enabling reuse of domain knowledge, making domain assumptions explicit, separating the domain knowledge from the operational knowledge, and analyzing domain knowledge.

On the other hand, there exist several arguments and challenges, among which are the lack of an agreed-upon taxonomy and quantitative evaluation procedures.

## 2.3 **SERVICE ORIENTED SENSOR NETWORKS**

Despite the amount of research devoted to ontology design and development, very little attention has been paid to semantic representation of wireless sensor networks data and its properties. OntoSensor includes definitions of concepts and properties adopted in part from SensorML, extensions to IEEE SUMO and references to ISO 19115. It presents a practical approach to building a sensor knowledge repository in a network-centric environment [9].

The idea of using ontology-driven information system for sensor networks is not entirely new. The work in [5] presents an attempt to capture the most important features of a sensor node that describes its functionality and its current state. The ontology describes the main components of a sensor node such as processor CPU and memory, power supply, and radio and sensor modules.

A step further in ontology-based sensor nodes is presented in [10] and [11]. The researchers in [12] define an ontology that integrates high level features that characterizes sensor networks for customizing routing behavior. The proposed ontology describes the network topology and settings, sensor description, and data flow.

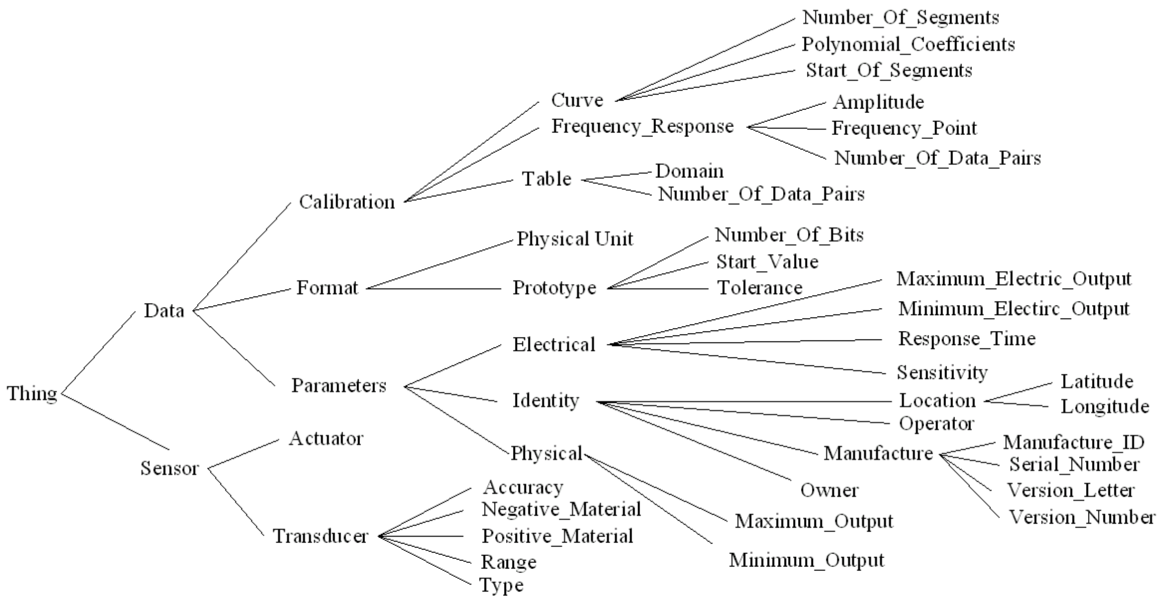


**Fig. 1.** Sensor data ontology

Again, there is no mention of sensor data. Subsequent work like [13] is an effort in the direction of facilitating semantic-service oriented sensor information systems. The notion of ontology used in this research is to capture the information about physical entities that sensors sense and their relationships.

The IEEE 1451 is a family of proposed standards that provide a single generic interface between a transducer and external network protocol in use [14]. The IEEE 1451 standard family uses Transducer Electronic Data Sheet (TEDS) to capture sensor characteristics, such as transducer identification, calibration, correction data, and manufacturer related information. That is, the approach of sensor ontology has designed to focus on physical versus electrical data of sensor information like Figure 1. Consequently, much of the knowledge captured by the ontology describes the widely accepted IEEE 1451 TEDS templates.

However, as mentioned before, this paper designs the novel sensor ontology based on service-oriented properties and enable compatibility with legacy sensor ontology for service-oriented service.

## III. Service Oriented Sensor Networks for Mobile Computing

### 3.1 **Service Domain**

Our main source for service-oriented service commonly used term in service domain is the ubiquitous environments. It is possible for user participate in networking at anytime and anywhere through portable devices under ubiquitous environments like Figure 2.

In this surrounding circumstance, the user can get the information which is provided from sensor nodes in WSNs immediately, and we have to think the following questions at this time.

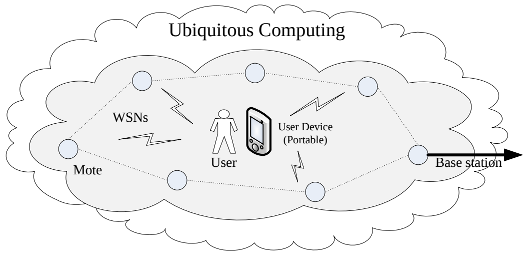


**Fig. 2.** Service domain

- Is providing information from sensor nodes what either sensor node ID or value of temperature like “25”
- And also is any additional available?, If so, it physical information?
- Where does the user utilize the physical information?

Generally, user expects the value of temperature like “25” for above questions and hopes that it will be displayed on portable device in addition. Therefore, service-oriented properties of this paper focused on service-oriented approach like Figure 3.

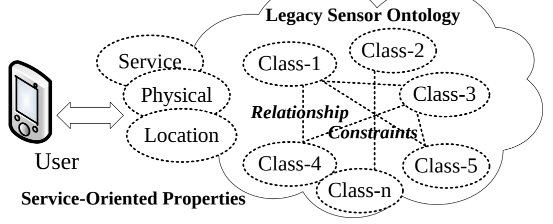


**Fig. 3.** Service-oriented approach

### 3.2 **Service-Oriented Properties**

The meaning of the service-oriented property is sensing data and is different from sensor data. ***Service***, ***Location*** and ***Physical*** showed in Figure 3 are service-oriented properties which are expected to the user. On the other side, similar properties relate to service-oriented properties can be extracted from the legacy ontology including great many information of the physical sensor and its property, but these are physical-oriented properties. For example, the temperature value "25" which is captured by a temperature sensor in the field is service-oriented property, but CPU type, memory size, sensing mechanism, battery power, actuator and transducer are physical-oriented data about the sensor.

## 3.3 **Ontology Design**

The ontology development follows an evolving prototype life cycle rather than a waterfall or an iterative one. This implies that one can go back from one stage to another stage in the development process as long as the ontology does not satisfy or meet all the desired requirements. Therefore, the usually accepted stages through which ontology is built are: collecting vocabulary commonly used, identifying an initial taxonomy, adding restrictions and axioms, consistency checking, incremental modifications, and evaluation .

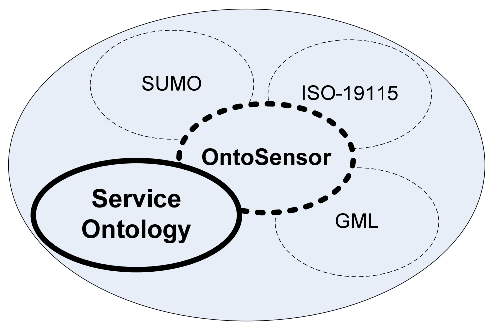


**Fig. 4.** Outline of the proposed ontology

Therefore, our base ontology for collecting commonly used terms in service domain are OntoSensor which import terms from OGC and SUMO ontology.

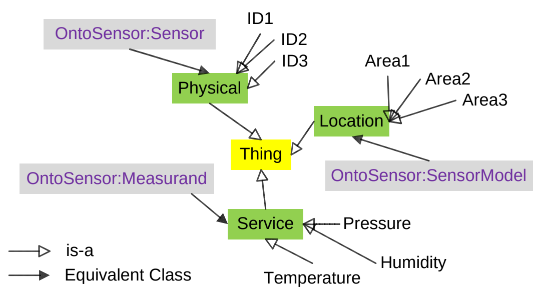


**Fig. 5.** Design of the proposed ontology

As importing this ontology, we designed the novel sensor ontology like Figure 4 and 5 in designing process. We defined the *Service*, *Location* and *Physical* class and linked the relationships with OntoSensor ontology using "*owl:equivalentClass*" property for service-oriented service.

"*owl:equivalentClass*" is a built-in property that links a class description to another class description. The meaning of such a class axiom is that the two class descriptions involved have the same class extension (i.e., both class extensions contain exactly the same set of individuals). That is, "*Service*", "*Physical*" and "*Location*" are equivalent class with "*OntoSensor:Measurand*", "*OntoSensor: Sensor*" and "*OntoSensor:SensorModel*" respectively. The next step is to take the list of concepts as described by the identified terms and form the initial class taxonomy. This implies looking at whether a concept is a sub-concept of another one or not. Figure 6 shows our initial taxonomy after adding a few dozen concepts. Concepts were added one at a time, structuring the taxonomy as needed to accommodate each concept.

## 3.4 **Properties and Constraints**

Relationships among classes are usually referred to as properties. A property links an individual from its domain to an individual of its range. Notice that the links from classes to their sub-classes represent properties that are listed in Table 1.

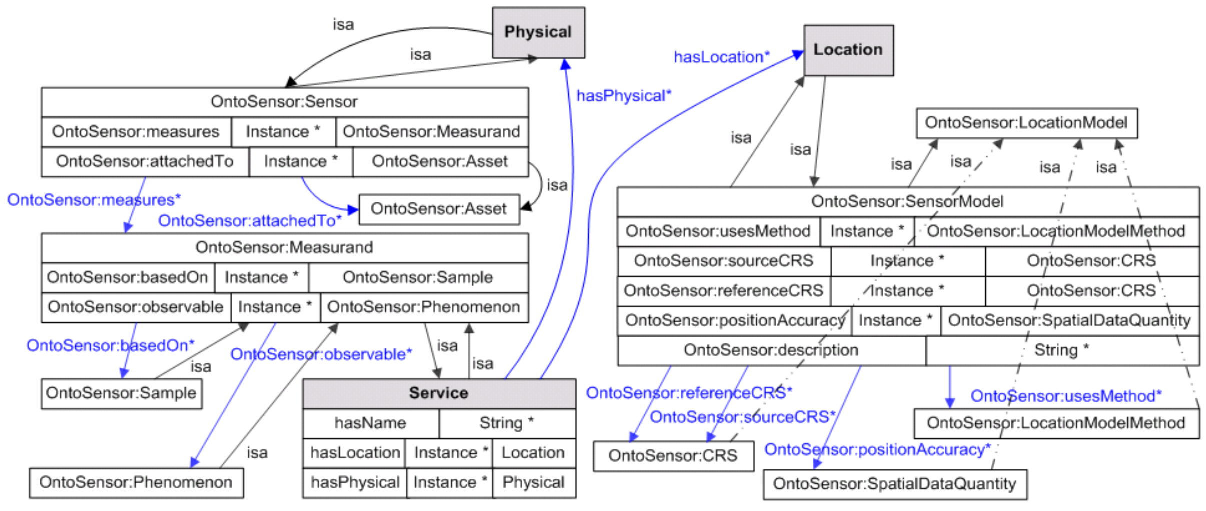
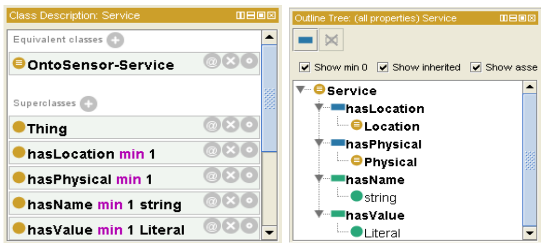

Physical
isa
isa
OntoSensor:Sensor
OntoSensor:measures | Instance * | OntoSensor:Measurand
OntoSensor:attachedTo | Instance * | OntoSensor:Asset
isa
OntoSensor:Asset
OntoSensor:measures*
OntoSensor:attachedTo*
OntoSensor:Measurand
OntoSensor:basedOn | Instance * | OntoSensor:Sample
OntoSensor:observable | Instance * | OntoSensor:Phenomenon
OntoSensor:basedOn*
OntoSensor:observable*
isa
isa
isa
OntoSensor:Sample
isa
OntoSensor:Phenomenon
Service
hasName | String *
hasLocation | Instance * | Location
hasPhysical | Instance * | Physical
hasPhysical*
hasLocation*
Location
isa
isa
OntoSensor:LocationModel
isa
isa
isa
isa
OntoSensor:SensorModel
OntoSensor:usesMethod | Instance * | OntoSensor:LocationModelMethod
OntoSensor:sourceCRS | Instance * | OntoSensor:CRS
OntoSensor:referenceCRS | Instance * | OntoSensor:CRS
OntoSensor:positionAccuracy | Instance * | OntoSensor:SpatialDataQuantity
OntoSensor:description | String *
OntoSensor:referenceCRS*
OntoSensor:sourceCRS*
OntoSensor:usesMethod*
OntoSensor:CRS
OntoSensor:positionAccuracy*
OntoSensor:LocationModelMethod
OntoSensor:SpatialDataQuantity

**Fig. 6.** Onto Sensor Architecture In This Paper

**<Table 1>** Properties list

| **Domain** | **Property Name** | **Range** |
|---|---|---|
| *Service* | *hasLocation* | *Location* |
| | *hasPhysical* | *Physical* |
| | *hasName* | *Service (OntoSensor:Measurand)* |
| | *hasValue* | *xsd:decimal* |
| *Location* | *hasLongitude* | *OntoSensor:LocationModel* |
| | *Description* | *xsd:string* |
| | *positionalAccuracy* | *OntoSensor:SpatialDataQuentity* |
| | *referenceCRS* | *OntoSensor:CRS* |
| | *sourceCRS* | *OntoSensor:CRS* |
| | *usesMethod* | *OntoSensor:LocationModelMethod* |
| *Physical* | *measures* | *OntoSensor:Sensor* |


Class Description: Service
Equivalent classes
Superclasses
Outline Tree: (all properties) Service
Show min 0
Show inherited
Show asse

OntoSensor-Service

Thing
hasLocation min 1
hasPhysical min 1
hasName min 1 string
hasValue min 1 Literal

Service
hasLocation
Location
hasPhysical
Physical
hasName
string
hasValue
Literal

**Fig. 7.** Class description: Service

For instance, the link from *Service* class to *Location* and *Physical* class represents the property "*hasLocation*" and "*hasPhysical*", respectively. And the *hasLocation* property links the *Service* class to either *Area1* or *Area2*, or *Area3* entities in *Location* class. Figure 7 shows the description of the "*Service*" class and its object property, range and data property.

## IV. Implementation and Validation

### 4.1 **Imported Ontology and Instance**

In this section, we present our technical judge of the designed ontology by performing the tests mentioned in section 4. The experimental evaluation is limited to validating the ontology (checking for logical inconsistencies) and querying the services.

Eventually, comparing the performance parameters of a search engine (such as precision, recall, and response time) when utilizing the ontology versus traditional searching (such as databases) is a vital part of the performance analysis.

Ontology(http://sslab.cheju.ac.kr/~kjh/ontology/upservice.owl)
Ontology(http://sslab.cheju.ac.kr/~kjh/ontology/OntoSensor.owl)

**Fig. 8.** Imported ontology

Therefore this performance testing is our immediate future work. Imported OntoSensor ontology for our novel proposed ontology showed in Figure 8.

Table 2 displays an excerpt of the OWL file generated as the output of the instantiation. The excerpt shows the OWL constructs that capture the following knowledge: The resource "Service" is an instance of the *Service* class. This instance of a service is appropriate for obtaining "Physical", has physical information of sensor, and can be used for projects that require some hardware information; hence the "Service" also has location information.

**<Table 2>** Excerpt of Service instance

```
<Service rdf:ID="Service 20">
  <hasName
rdf:datatype="&xsd;string">Humidity</hasName>
  <hasValue rdf:datatype="&xsd;float">60.0</hasValue>
  <hasPhysical rdf:resource="#Physical 16"/>
  <hasLocation rdf:resource="#Location 11"/>
</Service>
<Service rdf:ID="Service 21">
  <hasName
rdf:datatype="&xsd;string">Temperature</hasName>
  <hasValue rdf:datatype="&xsd;float">23.0</hasValue>
  <hasPhysical rdf:resource="#Physical 17"/>
  <hasLocation rdf:resource="#Location 12"/>
</Service>
<Service rdf:ID="Service 22">
  <hasName rdf:datatype="&xsd;string">CO2</hasName>
  <hasValue rdf:datatype="&xsd;float">78.0</hasValue>
  <hasPhysical rdf:resource="#Physical 18"/>
  <hasLocation rdf:resource="#Location 13"/>
</Service>
<OntoSensor-Phenomenon rdf:ID="OntoSensor-
Phenomenon 2">
  <hasName rdf:datatype="&xsd;string">Pressure</hasName>
  <hasValue rdf:datatype="&xsd;float">20.0</hasValue>
  <hasPhysical rdf:resource="#Physical 18"/>
  <hasLocation rdf:resource="#Location 11"/>
</OntoSensor-Phenomenon>
<OntoSensor-Sample rdf:ID="OntoSensor-Sample 1">
  <hasName
rdf:datatype="&xsd;string">Temperature</hasName>
  <hasValue rdf:datatype="&xsd;float">19.0</hasValue>
  <hasPhysical rdf:resource="#Physical 19"/>
  <hasLocation rdf:resource="#Location 14"/>
</OntoSensor-Sample>
```

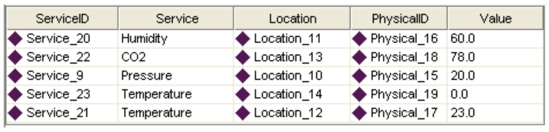
### 4.2 **Query Results**

As a query language, we used to SPARQL query language which is provided and plugged-in to Protégé. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. It also contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions, and also supports extensible value testing and constraining queries by source RDF graph.

**<Table 3>** SPARQL query statements

```
prefix
here:<http://sslab.cheju.ac.kr/~kjh/ontology/uServiceUp.owl#>
SELECT  ?ServiceID ?Service ?Location ?PhysicalID ?Value
WHERE {  ?ServiceID here:hasName ?Service .
         ?ServiceID here:hasLocation ?Location .
         ?ServiceID here:hasPhysical ?PhysicalID .
         ?ServiceID here:hasValue ?Value   }
```

The results of SPARQL queries can be results sets or RDF graphs. SPARQL is built on the triple pattern, which also consists of a subject, predicate and object. A triple from our data expressed using the SPARQL triple pattern syntax looks like this: ***<ontology URL> prefix:property value***. And we also use the triple pattern to include only variables like this: ***?subject ?predicate ?object***. Figure 11 showed the query results about temporary service query. When we are somewhere, we have following question: "What is it?", and "Where are we?". In here, that is a temperature service and our current location is the same location where physical sensor was deployed. Therefore, service query including location and physical information in proposed ontology can be make as Table 3.



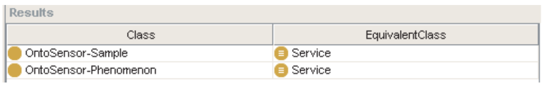
| ServiceID | Service | Location | PhysicalID | Value |
|---|---|---|---|---|
| ◆ Service_20 | Humidity | ◆ Location_11 | ◆ Physical_16 | 60.0 |
| ◆ Service_22 | CO2 | ◆ Location_13 | ◆ Physical_18 | 78.0 |
| ◆ Service_9 | Pressure | ◆ Location_10 | ◆ Physical_15 | 20.0 |
| ◆ Service_23 | Temperature | ◆ Location_14 | ◆ Physical_19 | 0.0 |
| ◆ Service_21 | Temperature | ◆ Location_12 | ◆ Physical_17 | 23.0 |

**Fig.11.** Temporary query results

On the other side, it is possible to search the service-oriented properties in OntoSensor ontology because service-oriented classes of the proposed ontology has been defined as equivalent relationships to service-oriented classes of the OntoSensor ontology which is similar to proposed ontology but physical-based. Figure 12 shows the service-oriented classes searching results among classes which are in proposed ontology and OntoSensor ontology using SPARQL query like Table 4. Figure 13 is the results of the re-query using SPARQL query in Table 3.

**<Table 4>** Searching equivalent classes

```
prefix
here:<http://sslab.cheju.ac.kr/~kjh/ontology/uServiceUp.owl#>
SELECT  ?Class ?EquivalentClass
WHERE  { ?Class rdfs:subClassOf ?Object .
         ?Object owl:equivalentClass ?EquivalentClass }
```

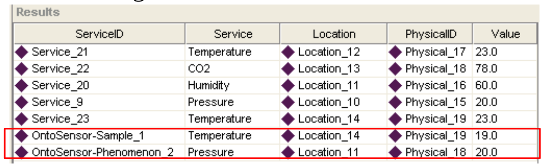


Results

| Class | EquivalentClass |
|---|---|
| ● OntoSensor-Sample | ● Service |
| ● OntoSensor-Phenomenon | ● Service |

**Fig. 12.** Results of Searching equivalent classes

Therefore, we knew that information related to equivalent classes which are marked red box in Figure 13 are searched and added to Figure 11.



Results

| ServiceID | Service | Location | PhysicalID | Value |
|---|---|---|---|---|
| ◆ Service_21 | Temperature | ◆ Location_12 | ◆ Physical_17 | 23.0 |
| ◆ Service_22 | CO2 | ◆ Location_13 | ◆ Physical_18 | 78.0 |
| ◆ Service_20 | Humidity | ◆ Location_11 | ◆ Physical_16 | 60.0 |
| ◆ Service_9 | Pressure | ◆ Location_10 | ◆ Physical_15 | 20.0 |
| ◆ Service_23 | Temperature | ◆ Location_14 | ◆ Physical_19 | 23.0 |
| ◆ OntoSensor-Sample_1 | Temperature | ◆ Location_14 | ◆ Physical_19 | 19.0 |
| ◆ OntoSensor-Phenomenon_2 | Pressure | ◆ Location_11 | ◆ Physical_18 | 20.0 |

**Fig. 13.** Query results

## V. Conclusion and Future Work

Mobile computing is a relatively new field. While the challenges arising from mobility and the limitations of the portable devices are relatively well understood, there is no consensus yet as to what should be done to address these challenges. A comprehensive solution has to address many different aspects, such as the issue of dynamically changing bandwidth, the power, computational, and other limitations of the portable devices, or the varying availability of services in different environments.

The semantic representation of wireless sensor networks data in mobile computing is an exciting vision that enables structured information to be interpreted unambiguously. Precise interpretation is a necessary prerequisite for automatic search, retrieval, and processing of sensor data for mobile computing. For ubiquitous mobile computing, this paper is the first attempt to define ontology for describing concepts and relationships of the wireless sensor networks based on service-oriented service.

The benefits of this paper are to classify the property and provide the new approach about the wireless sensor networks ontology for service-oriented service in the future ubiquitous computing and are the approach based on service properties rather than physical properties.

As for future work, we are considering extending the ontology so that it describes the entire available property base on service preference; including URL and UFID location property. Moreover, we plan to test the effectiveness of the ontology approach by quantitatively measuring the improvements in comparisons with legacy ontology and recall rates of a search engine when utilizing the ontology against traditional string-based searching approaches and also make the killer application using proposed ontology and develop the embedded programming technology. This effort will be a further step in the direction towards enabling ubiquitous services to access and process sensing and sensors data.

**Acknowledgments**

This work (Grants No.C0024433) was supported by business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2012.

## References

[1] M. Othman and S. Hailes, *Power Conservation Strategy for Mobile Computers Using Load Sharing*, Mobile Computing and Communications Review, 2(1), pp.44-51, January 1998.

[2] D. J. Russomanno, C. Kothari and O. Thomas. Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. *The 2005 International Conference on Artificial Intelligence*, Las Vegas, NV, pp. 637-643, 2005.

[3] S. Avancha, C. Patel, and A. Joshi. Ontology-driven Adaptive Sensor Networks. *First Annual International Conference on Mobile and Ubiquitous Systems, Networking and Services*, pp. 194-202, Aug., 2004.

[4] H.-Y. Lo, *M-Mail: A Case Study of Dynamic Application Partitioning in Mobile Computing*, Master's Thesis, Department of Computer Science, University of Waterloo, May 1997.

[5] Mohamad Eid, Ramiro Liscano, Abdulmotaleb El Saddik. A Universal Ontology for Sensor Networks Data. *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, Ostuni- Italy, pp. 59-62, June 2007

[6] T. D. Hodes, R. H. Katz, E. Servan-Schreiber, and L. Rowe, *Composable Ad-hoc Mobile Services for Universal Interaction*, Proc.3rd Ann. ACM/IEEE Conf. on Mobile Computing and Networking, Budapest, Hungary, pp. 12.September 1997.

[7] Ontology Editor and Knowledge Acquisition System, http://protege.stanford.edu/, accessed Oct., 2008.

[8] Racer Systems web site, http://www.racersystems.com/, accessed Oct., 2008.

[9] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/, accessed Oct., 2008.

[10] Leigh Dodds. Introducing SPARQL: Querying the Semantic Web. http://www.xml.com/pub/a/2005/11/16/int-roducingsparql-querying-semantic-web-tutorial.html, accessed Oct. 20, 2008.

[11] Nicola Guarino, Massimiliano Carrara, Pierdaniele Giaretta. An Ontology of Meta-Level Categories. *Proceedings of the Fourth International Conference (KR94)*, Morgan Kaufmann, San Mateo, CA, pp. 270-280 1994.

[12] Natalya F. Noy and Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*, March 2001.

[13] R. Jurdak, C. V. Lopes, and P. Baldi. A Framework for Modeling Sensor Networks. *Proceedings of the Building Software for Pervasive Computing Workshop at OOPSLA'04*, Vancouver, Canada, Oct., 2004.

[14] George V. Cybenko, Guofei Jiang, and Wayne Chung. Semantic Agent Technologies for Tactical Sensor Networks. *Proceedings of the SPIE Conference on Unattended Ground Sensor Technologies and ApplicationsV*, Orlando, FL, SPIE, pp. 311-320, Sep., 2003.

[15] Jie Liu and Feng Zhao. Towards Semantic Services for Sensor-Rich Information Systems. *Proceedings of the 2nd IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks (Basenets 2005)*, Boston, MA, pp. 967-974, Oct., 2005.

[16] IEEE P1451Project official web site, http://ieee1451.nist.gov, accessed in 27, Sep., 2007.

[17] M. Botts, A. Robin. OpenGIS Sensor Model Language (sensorML) Implementation Specification. http://www.opengeospatial.org, accessed Oct., 2007.

[18] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe. A Practical Guide to Building OWL Ontologies Using the Protege-OWL, Plugin and COODE Tools, Edition 1.0. *Tutorial*, http://www.coode.org, accessed Oct., 2007.