A Composite Sensor-based Context Modeling Method for Context-aware Pervasive Computing

XU Yusheng, MA Zhixin†, CHEN Xiaoyun, LI Lian

***Abstract*—Context-aware systems are one of the most important application domains of pervasive computing. In this paper, a new context model is proposed, which is based on the composite sensor concept. Unlike simple ones (physical, virtual or logical sensors), a composite sensor is a logical sensor which is integrated from simple and/or other composite ones. Therefore, composite sensor has more semantic meaning than simple ones and can be used to describe complicated context domain objects, in a natural way that separates requirement from design. Based on composite sensor context model, we present a context modeling method, ComSensor, and a pervasive computing architecture, ComArch, which is suitable for different kinds of pervasive computing applications. The performance examining results of a simulation prototype system show that applications based upon ComArch can run in real-time mode.**

***Index Terms*— context-aware system, context model, distributed computing, pervasive architecture, pervasive computing.**

I. Introduction

With the increasingly development of wireless network technology and mobile devices such as mobile phones, PDAs, etc., pervasive computing is becoming a focused research area these days and attracts the most attention. Pervasive computing is firstly introduced by M. Weiser [13] in 1991 which refers to the computing system that is seamless integration of devices into the users' everyday life environment. "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." Pervasive computing has broad application areas such as location aware applications, patient healthcare, smartness office, intelligent conference room, etc.

One field in the wide range of pervasive computing is the so-called context-aware system, which may adapt its operations to the current context without explicit user intervention. Context data/information is defined as the current location, time, light, sound, movement, touch, temperature, air pressure, object and object status or other environment attributes which are related to the application domain. Context data is usually collected through sensor-devices and processed for providing usable and effective services with taking environmental context into account [1].

Take smartness office [12] as an example, in the pervasive computing environment, users' current locations can be identified by the mobile devices they wore or carried (e.g., cell phones, PDAs, badges or headphones). Then several context-aware services can be supported such as forwarding phone calls to a telephone close to the user based on his current location, temperature or light controlling when a user moves in, and so on. For pervasive healthcare [11], consider another example. A possible scenario is to store all critical medical information (e.g., blood group, allergies, disease symptoms, medical records) on a patient's mobile device. When she moves into the office of a doctor, the medical information is uploaded to the doctor's computer for further diagnosing. And when she moves out of the door, all the updated information is downloaded to her mobile device for future usage.

Constructing a pervasive computing system has two sub-problems which are context model and pervasive architecture. Context model correlates with building a domain semantic context objects model, while the pervasive architecture provides a real-time framework for the context-aware services. This context model is captured from special application domain and used by developing users. Thus, the context model should be depicted in a natural way and developers can design context model with the assisting of semantic modeling method. The requirement for a pervasive architecture is detecting context objects and providing context-aware services in real-time mode, no matter how many types of context information.

The task of constructing a pervasive computing system is quite challenging. For example, there are so many different types of context objects in the domain environment and they are very different from one another. Thus context object modeling is an expensive task. As more and more types of mobile devices and context objects are considered while the system should run in real-time mode, the scale-up and others are difficult issues. The overall pervasive computing architecture designing may be a nerve-wracking task for developers. Our basic idea is to integrate different related sensors about one context object into a conceptual one. With different detail abstract levels, context modeling is simplified.

In this paper, we proposed a composite sensor-based context model, which may integrate context data and data processing. Composite sensor-based model provides a method for designing in different abstract level, which is very

Manuscript received November 28, 2007. This work was supported in part by the National Natural Science Foundation of China under Grant No.90612016 and Grant No. 60473095.

XU Yusheng is with School of Information Science and Engineering, Lanzhou University, 730000, China (e-mail: xuyusheng@ lzu.edu.cn).

† Corresponding author. MA Zhixin is with School of Information Science and Engineering, Lanzhou University, 730000, China (phone: 86-931-8910278 fax: 86-931-8910278; e-mail: mazhx@lzu.edu.cn).

CHEN Xiaoyun is with School of Information Science and Engineering, Lanzhou University, 730000, China (e-mail: chenxy@lzu.edu.cn).

LI Lian is with School of Information Science and Engineering, Lanzhou University, 730000, China (e-mail:lil@lzu.edu.cn)

useful for building a model for different users. Taking this context model as cornerstone, a uniformed pervasive architecture is presented. This architecture is scalability, flexible, extensible and is appropriate for building pervasive application systems.

This paper is structured as follows. Section 2 introduces the background of pervasive computing. Related works are described in section 3. Section 4 proposed the composite sensor and the related context model. The context modeling method and comparing analysis can be found here, too. And in section 5, based on composite sensor-based context model, a pervasive architecture is presented which is suited to implement pervasive applications. Section 6 presents the prototype and simulation result. Finally, conclusions can be found in section 7.

II. BACKGROUND

Pervasive computing is the evolutionary result of distributed systems and mobile computing. Although they are different computing styles, distributed systems and mobile computing are the foundation for pervasive computing. Context dependency is a major issue in pervasive computing which is a specialization of mobile, distributed systems. The relationship is outlined in Fig. 1.

Context-aware systems are highly related to the problem domains, and different domain or topic needs different solution. Flexible context data expression and robust architecture designing are the most challenging tasks for all systems. This section firstly gives a general description of context model for context information representing, storing and exchanging. Furthermore, basic requirements for architecture designing are analyzed.

A. Context Model

Context refers to any information that can be used to characterize the situation of objects that are considered relevant to the interaction between a user and an application [1]. While context model is used to express context object, data structure, relationship, rule, information integration and other focused attributes. Context information has many characteristics such as distributed, partial validated, incomplete or ambiguous, integrated, etc. As a context information expression method, context model should be simplicity, flexibility, extensibility, generalized and expressiveness [1, 8].

Given a domain context, developing a flexible context model that covers a wide range of possible context objects is the key for implement the context-aware system. Other issues such as data storing, processing, exchanging can be easily resolved when we have a good designed context model.

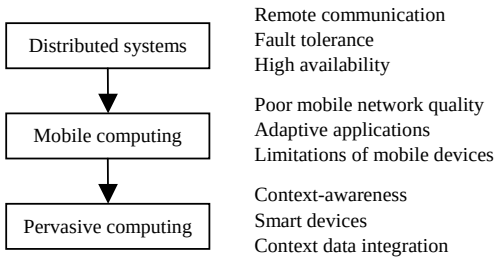


Figure 1. **Evolution of pervasive computing**

B. Basic Architecture Requirements

System architecture is the framework of the pervasive computing application. Because of different domain requirements and characteristics, the architectures of pervasive computing systems were implemented in many ways, which depend upon properties and conditions of devices, users, and extensibility. Context model is another key factor for architecture designing. The most common approach for pervasive computing frameworks is a classical hierarchical infrastructure with one or many centralized components using a layered architecture.

When design a pervasive computing architecture, several general requirements needed to be considered. Some of them are: (1) Integrating different types of devices for context sampling and detecting to collect context data. (2) Supporting spontaneous context-aware ad hoc communication among related devices and users. (3) Context aggregation or interpretation to process raw context data into desirable information. (4) High level of security and privacy to protect sensitive information about people. (5) Other good software quality criterions such as extensibility, robustness, simplicity, flexibility, etc. should be met, too.

III. RELATED WORKS

Since M. Weiser [13] in 1991 introduced pervasive computing, several systems have been implemented, such as smartness office [12], Aura [4], Intelligent Room [3], etc. Among the design tasks, context model and pervasive computing architecture are the two most studied topics.

In the previous works on context model, several modeling approaches were proposed, e.g., key-value model, markup scheme model, graphical model [9], object oriented model [6], logic based model and ontology base model [10]. Among all these modeling approaches, ontology-based methods independently proposed by H. Chen et al. [2] and Tao Gu et al. [2] are the most expressive one. Ontology represents a description of the concepts and relationships, and thus is a very promising instrument for modeling contextual information. Unfortunately, all these methods are based on simple single physical or logical/virtual sensor, unlike our composite sensor model, which can integrate simple sensors into a complex one.

As to pervasive architecture, a common framework in used is the hierarchical infrastructure, such as Context Managing Framework [7], Context Broker Architecture (*CoBrA*) [2], *ContextServer* [6], etc. Context Managing Framework [7] uses a three layers architecture, which includes Application, Context Manager and server or service layer. By introducing

broker middleware, *CoBrA* [2] has a three-layer architecture, too. The three layers are handheld computers, *CASS* (Context-awareness sub-structure) middleware and sensor nodes. The three layers of *ContextServer* [6] are application layer, management layer and adaptor layer. All of these frameworks are related to their accordingly pervasive environment or project respectively. In [1], M. Baldauf et al. proposed a more general multi-layered architecture, which is presented in Fig. 2. Having five function independent layers, this architecture is appropriate for large complex projects and

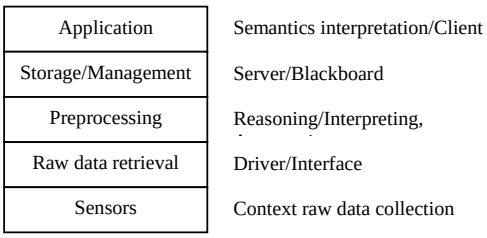


Figure 2. **Multi-layered conceptual architecture**

can be tailored for simple ones. But its high complexity may have a bit more overhead.

IV. Composite sensor-based context model

In this section, some notations are defined for simplify discussion. Then, the composite sensor-based context model and modeling process are proposed. Lastly, comparing analysis with other context model methods is made.

A. Basic Notations

Physical sensors. The most frequently used types of sensors are physical ones in pervasive computing environment. These hardware sensors can detect almost any raw context data, such as light, audio, location, temperature, etc. For example, badge, mobile phone/ PDA, GPS, etc. are all physical sensors which are carried or wore by users in location-based pervasive service systems.

Virtual sensors. Virtual sensors collect context data from other software applications or services. The raw context data is collected indirectly from physical sensors. For example, the temperature raw data in a room can not only be detected by using a thermometer (a physical sensor) but also be read from an air conditioning system. Although the temperature may be detected by physical sensors in air conditioning system, the air conditioning system is a virtual sensor for pervasive computing environment in this scenario.

Logical sensor. Logical sensors make use of a couple of data sources, such as physical sensors, virtual sensors, databases, etc. in order to solve higher tasks. For example, a logical sensor can be constructed to detect the current position of an employee by analyzing logins at desktop PCs and a database mapping of devices to location information. That is to say, logical sensor detects context raw data through events occurred in the system rather than by physical sensors.

B. Composite Sensor

All types of sensors mentioned above are simple, because they can only detect a single raw data. Relationships among different types of context data or results of combined data are unknown. In order to get these types of information, other processing should be carried out, which are separated from context data. For example, the pulse rate of a patient can be detected by a physical sensor. But weather it is in the normal scope can only be gotten by additional comparing operations. Modeling context objects by simple sensor concept is an uneasy way, and may impact context architecture designing. Few context model approaches proposed previously define these issues.

The concept of composite sensor is recursively defined as follows:

(1) A simple is a composite sensor.

(2) Adding additional operations upon a composite sensor may get another composite sensor.

(3) By defining the relationships of two composite sensors can get a new composite sensor.

(4) A new composite sensor can be gotten by integrating two related ones.

Operations, relations and integrating are related to special context pervasive application.

Take location-aware ubiquitous computing as an example. In this environment location information is a context entity. No matter where a person is currently in, his/her location must be automatically detected and calculated by a location sensor. But, by far as we know, there is no one simple sensor can detect all kinds of location raw data such as indoor, outdoor (campus or countryside). Depended on the current location of a person, different sensors may be used. For example, GSP may be used to countryside location detect, mobile phone can be used to detect location in the scope of a city, and so on. In this situation, composite sensor may help to model the context. Fig.3 shows the composite sensor example. GPS, mobile phone and campus sensor are physical sensors which can be used to detect the current location of persons in countryside, a city or campus. They are simple sensors and all can detect location information. By setting priority or comparing accuracy, these simple sensors can build a composite sensor – outdoor location sensor, which can tell the currently location when a person goes out of building. Similarly, simple sensors (*RDF* sensor, mobile phone and carried badge) build an indoor location composite sensor. Recursively, these two composite sensors may construct a new composite sensor- location sensor. By integrating all these physical or composite sensors together, the location sensor can detect location information no matter where the person is now in. This is the contribution of the concept of composite sensor.

A composite sensor is different from simple ones in many ways. (1) A composite sensor may be built from any number of simple ones. (2) The operations upon sensor values are encapsulated inside the sensor. (3) The relationship among sensors can be integrated into a new composite sensor, which provides more abstract conception. Based on simple ones, composite sensor concept is closer to the problem domain and thus supports an easy way to model the context objects.

C. Composite Sensor-Based Context Model

Taking composite sensor as a basic element (basic design element), composite sensor model we proposed can model the application context, which is defined as follows and depicted in formula (1).

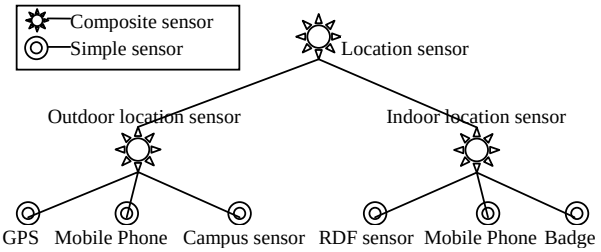


Figure 3. **Examples of composite sensor model**

$$ComS \equiv < Object, Context, Event, Rule > \quad (1)$$
$$Object \equiv < O, OA, RO >$$
$$Context \equiv < C, RC >$$
$$Event \equiv < EP, DF, DD >$$
$$Rule \equiv \{R, SPP\}$$

In formula (1), a composite-sensor (*ComS*) model is defined as a 4-ary which includes *Object*, *Context*, *Event* and *Rule*. In this formula, an *O* is the context object or entity which is the interest target. *OA* defines the attributes for the target. And, *RO* depicts the relationship among context objects. For example, in location-aware pervasive systems, person may be the context object, which has attributes (name, position, etc.). "Boss/manager of" is a relationship among persons. The *C* is the definitions of context and *RC* is the binary relationships between concepts in the context. Take location detecting system as an example, the context can be defined as campus, buildings and rooms. "Locate in" is binary relationship between buildings and campus or rooms and buildings. *EP* is the event that happens in the context and the flow that processes these events. *DF* represents data flow, which describes the flow path that how raw context data is converted into context information. *DD* describes context data and information distribution among system nodes. *R* expresses reasoning rules which decide how to provide location-based services. Lastly, *SPP* is the set of security/privacy policy rules.

In this context model, the raw context data is captured by composite sensors. Fig. 4 demonstrates an example in location-based pervasive computing. This simple example describes a context of a campus, which has a few of buildings. *Person* is the target object, whose attributes include *ID*, *Name*, *Position*, etc. The relationships among person objects are *BossOf*, *ManagerOf* and *Colleague*. As to the example, the context is constructed with *Campus*, *Building* and *Room* and the relation is *LocateIn*(*LocateInCampus*, *LocateInBuilding*).

The events in this example are *MoveInBuilding*, *MoveInRoom*, *GoOutBuilding* and *GoOutRoom*. Event processing flow, data flow and data distribution are omitted. Location-based service (reasoning rules) is modeled as "(*MoveInRoom,Dark,LightON*)", which means that when a person moves in a room and it is dark in the room, the light will be switched on automatically.

Additionally, the security and privacy issues may be taken into account when we model the context. For example, for a location query "where is Alice?", different abstract level of location information may be produced, such as "in building", "in campus" and "unknown" and "in Room204". As to the example in Fig.4 (*SPP*), if the query is posted by her manager, the response is "in Room204". If the query is requested by her colleagues, the result is "in laboratory building". And "unknown" is the result for others.

Except for data flow and data distribution, all content in this model are related to requirements. Composite sensor integrating, data flow and data distribution are issues related to system design. Thus, our proposed context model can separate requirement and system design.

```
<<{Person},{ID,Name,Position,…},{BossOf, ManagerOf,
   Colleague,…}>,
 <{Campus,Building, Room}, {LocateInCampus, LocateInBuilding}
   >,
 <{MoveInBuilding,GoOutBuilding, MoveInRoom,
   GoOutRoom,…},{…},{…}>,
 <{(MoveInRoom,Dark,LightON), (Owner,BeforeDoor,Open),
   (MoveInRoom,Cold,Heat),…},{(Manager,CurrentLocation ),
   (Colleague, Building), (Other,Unknown)…}>
>
```

Figure 4. **An example of composite sensor model (part)**

D. Composite Sensor-Based Context Modeling Process

Having composite sensor concept and model as foundation, a novel context modeling method named as *ComSensor* is proposed. The main steps of *ComSensor* method are described as follows.

(1) Identifying context target objects. Define their attributes, relationships.

(2) Defining context and the relations. Simple sensors are defined.

(3) Functional requirements analysis. Identify reasoning rules.

(4) Defining the composite sensors. Define data attributes, operations, relationships of all composite/simple sensors.

(5) Integrating all local model views into a global composite sensor model.

(6) Reviewing the global composite sensor model against overall goals of target system.

E. Comparing Analysis with Other Models

Ontology, markup scheme and UML are currently most used context model methods [1]. In Table I we compare our composite sensor-based context model (denoted as *CSB* context model) with these model methods. The features used for comparison are defined by Korpipää et al. [7] and valuable model requirements.

We can easily conclude from Table I that composite sensor-based context model outperforms others except for extensibility and expressiveness.

V. Composite Sensor-Based Pervasive Architecture

This section describes the uniform pervasive computing architecture (*ConArch*) which is based on composite sensor model. Firstly, the overall architecture is presented. Further more, how pervasive computing problems are resolved is described.

Table I. **Context models comparison**

Feature \ Model	CSB	Ontology	Markup	UML
Simplicity	G	G	G	A
Flexibility	G	G	G	G
Extensibility	A	G	G	A
Generality	G	G	A	A
Expressiveness	A	G	A	A
Security and privacy	G	U	U	U
Composite sensor	G	U	U	U
Reasoning rules	G	A	A	A
Separate requirement from design	G	A	U	U

Note: G - good, A - available, U – unknown

A. *The layered Overall Architecture*

Based on composite sensor concept and model mentioned above, a uniform pervasive computing architecture is proposed, named as *ComArch*, which is shown in Fig. 5. As most common distributed context-aware framework design approaches, this architecture uses a layered infrastructure, too. The four layers are simple sensors, composite sensors and services, management, and applications layer. Additionally, security and data storage issues are resolved in security and context object database modules.

The first layer consists of a collection of different types of simple sensors. Raw data capturing methods or data sources are packaged into this layer. All simple sensors are connected by pervasive ad hoc network. The second layer is composite sensors, which responsible for raw data retrieval, preprocessing and implementing functions (operations) of composite sensors. It makes use of appropriate drivers or APIs of the first layer, and provides an event-based APIs for upper layers. Each composite sensor keeps its current status data and optional segmental history data. Some common services such as location mapping are also implemented in this second layer. The third layer, management, is divided into two parts: sensor manager and event-based APIs. Sensor manager maintains the composite sensors in a uniformed way. It includes sensor defining, configuring, and so on. The database interface is implemented in this module. Event-based APIs provides an immediate interface for upper applications. By using immediate APIs or database access methods, the fourth layer of *ComArch* implements the actual reaction on different events and context instances. It is the interface between pervasive computing systems and client users. The event-based immediate APIs are used for automatically pervasive services, while database access methods are used for pervasive queries posted by users.

B. *ComArch and Context Model*

The elements of context model defined in section 4 are integrated in *ComArch*. Context objects are implemented as composite sensors. Having encapsulated all details of a context object (current status data, attributes, relationships, rules and event processing algorithms), composite sensors can make decision based on reasoning rules which are defined in context model. Context and context relationship are encapsulated in common services in the third layer. Data flow and data distribution defined in context model will determine the nodes distribution in pervasive ad-hoc network. Based on security and privacy policy, the security module guarantees only privacy rules governed context information can be response to user query.

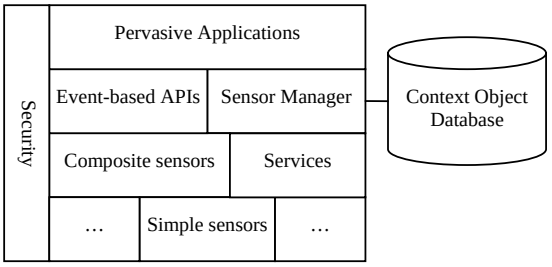


Figure 5. **The pervasive computing architecture *ComArch***

C. *Real-Time Feature in ComArch*

The characteristics of pervasive applications are real-time context sensor status retrieval, real-time system response, wide area information access, different types of client devices and sensor devices, optional large number of data transport. In the third layer of proposed pervasive architecture *ComArch* is a collection of event-based immediate APIs, which encapsulate immediate interfaces for composite sensor status retrieval. These immediate interfaces help *ComArch* architecture to guarantee the real-time features of pervasive services. As to the simulation results, *ComArch* can run in real-time mode.

VI. SIMULATION PROTOTYPE

In order to validate the effectiveness of *ComArch* architecture, we build a prototype system for location-notification. In this prototype, a user will receive notification about his current location, and he can also request current location of others. The location sensors are simulated by C++ threads. Users are notified through mobile phones, PDAs or PCs. All composite sensors and common services reside in a central server. These devices are connected together by wireless local network or Internet & GSM wide area public wireless network. The topology structure is shown in Fig. 6.

The developing environment for the prototype is as follows. The central server is built by an Intel Pentium 4 CPU 1.7GHz PC with 512MB main memory, running Microsoft Windows 2003 server and MySQL, with C++ compiling environment. The modules for location information demonstration on PDA or PC client are programmed in J2ME, while notification for mobile users is done through short message (SMS).

In the performance experiments, we record the delay of context events and location query requests. We measure delay of context events as time difference between events occurring and user client receiving notification. Delay of query is measured as time difference between request posting and receiving response. The prototype runs with 1000 composite sensors, which represents 1000 context objects (persons) in the context.

During the simulating, each person (simulated by a thread) walks through a path generated at random, and his current location is reported to the central server, then the server forwards location information to the client. After prototype running, a set of average sample data is generated for each 10 minutes. The simulation results are depicted in Fig.7.

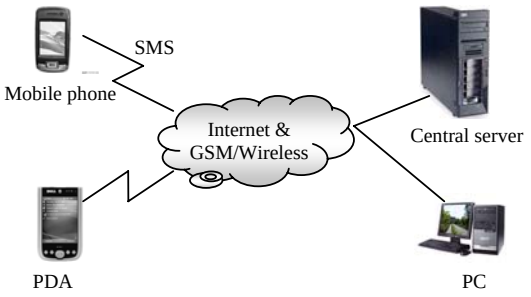
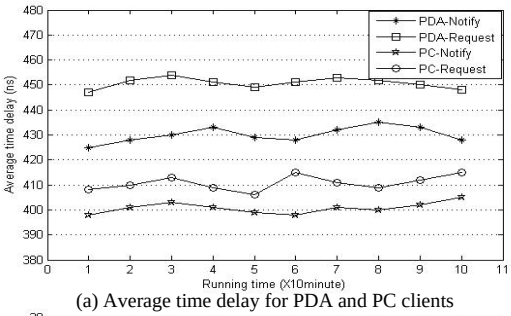
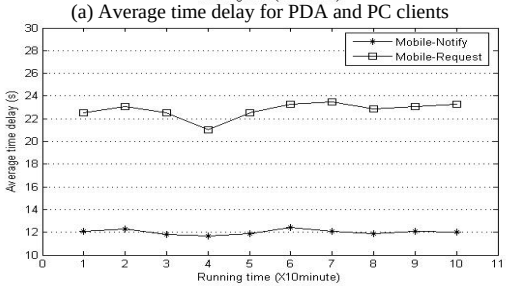


Figure 6. **Topology of the prototype system**



(a) Average time delay for PDA and PC clients



(a) Average time delay for mobile phone clients

Figure 7. **Performance of ComArch prototype**

Fig. 7 shows the average time delay for PDA, PC and mobile phone clients, including both current location notification service (labeled PDA-Notify, PC-Notify and Mobile-Notify) and location request query (labeled PDA-Request, PC-Request and Mobile-Request). The figure clearly indicates that the time delay for PDA and PC clients is no more than 460ns, while time delay for mobile phone client is no more than 24s, and *ComArch* runs in real-time mode. In addition the performance of notification service is faster than location request queries. The simulation system runs stably. There is no clearly performance decrease as time goes on.

Mobile phone clients connect server through short message (SMS). By checking short message gateway log file, we found that all most all time is spent on GSM network. 24s does not exceed time limitation of SMS application (30s). This is why PDA and PC clients outperform mobile phone clients. Current location information is pushed to client as soon as server catches composite sensor event through event-based APIs, while location query is requested from client, processed by central server and the result is sent back to the client. Thus, performance of location notification service is higher than that of location request.

VII. Conclusions

In this paper, we proposed a novel context model based on composite sensor concept. Comparing to previous works, it models context environment in a natural way. By separating context requirement capturing from system designing, all context-aware application requirement can be captured in this context model. Based on composite sensor context model, a uniformed pervasive architecture, *ComArch*, is proposed, which is appropriate for different kinds of pervasive computing environment. The simulation prototype system shows that *ComArch* architecture can support real-time performance.

References

[1] M. Baldauf, S. Dustdar and F. Rosenberg, “A survey on context-aware systems”, International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC), Volume 2, Number 4, 2007, pp.263-277.

[2] H. Chen, T. Finin and A. Joshi, "An ontology for context-aware pervasive computing environments", The Knowledge Engineering Review, Volume 18, Issue 3, 2003, pp.197 - 207.

[3] M. Coen, "The Future of Human-Computer Interaction, or How I learned to stop worrying and love my Intelligent Room", IEEE Intelligent Systems, 4(3-4), March/April 1999, pp. 8-10.

[4] D. Garlan, D.P. Siewiorek, A. Smailagic and P. Steenkiste, "Project Aura: toward distraction-free pervasive computing", IEEE Pervasive Computing, Volume 1, Issue 2, Apr-Jun 2002, pp. 22- 31.

[5] T. Gu, X. H. Wang, H. K. Pung and D. Q. Zhang, "An Ontology-based Context Model in Intelligent Environments", In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, San Diego, California, USA, January 2004, pp.270-275.

[6] Hofer, T. Schwinger, W. Pichler, M. Leonhartsberger, G. Altmann, J. and Retschitzegger, W, "Context-awareness on mobile devices - the hydrogen approach", Proceedings of the 36th Annual Hawaii International Conference on System Sciences-2003, Jan. 2003, pp.292-302.

[7] Korpipää P., Mäntyjärvi J, Kela J, Keränen Heikki and Malm E-J, "Managing context information in mobile devices", IEEE pervasive computing, Volume 2, Number 3, 2003, pp.42-51.

[8] M. Satyanarayanan, "Pervasive computing: vision and challenges", IEEE Personal Communications, Volume 8, Issue 4, Aug 2001, pp.10-17.

[9] Q.Z Sheng and B. Benatallah, “ContextUML: a UML-based Modeling Language for Model-driven Development of Context-aware Web Services”, in Proceedings of the International Conference on Mobile Business (ICMB’05), pp.206-212, 2005.

[10] T. Strang, C. Linnhoff-Popien, "A context modeling survey", UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management, Nottingham (2004), 2004, pp.34-41.

[11] U. Varshney, "Pervasive Healthcare and Wireless Health Monitoring", Mobile Networks and Applications, Volume 12, Numbers 2-3, 2007, pp.113-127.

[12] R. Want, A. Hopper, V. Falcão and J. Gibbons, “The Active Badge Location System”, ACM Transactions on Information Systems (TOIS), Volume 10, Issue 1, January 1992, pp. 91-102.

[13] M. Weiser, “The Computer for the 21st century”, Scientific American, 265(3), 1991, pp. 94-104.