International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.2 (2014), pp.309-318
http://dx.doi.org/10.14257/ijmue.2014.9.2.31

# Design and Embodiment of Web of Things Platform

Woo-Chang Shin

*Dept. of Computer Science, at SeoKyeong University*
*16-1 Jungneung-Dong Sungbuk-Ku Seoul, 136-704, KOREA*

*wcshin@skuniv.ac.kr*

***Abstract***

*Enormous efforts to build ubiquitous computing environment are made by connecting smart things, which are gradually increasing in real world, with network. The existing mode to build in the type of proprietary and tightly-coupled systems requires plenty of costs and time. Also, alteration of the already built system's structure is difficult. This paper proposes the Web of Things (WoT) platform to build ubiquitous computing environment. The proposed WoT platform enables to reduce costs required for building ubiquitous computing environment through the standardization of access to various smart things and control mechanism. The WoT platform also enables non-experts to easily build ubiquitous environment. To verify the effectiveness of the WoT platform, this paper develops a WoT platform prototype and carries out performance evaluation on the functional access to smart things.*

***Keywords:*** *Ubiquitous Computing, Web of Things, REST, WoT Platform, Smart things*

## 1. Introduction

Social needs for ubiquitous computing environment, in which anyone can easily, conveniently and safely receive IT service through computer and network anywhere, anytime increase, as IT technology develops [10, 14]. Typical ubiquitous computing-applied industries encompass LBS (location-based services), U-health care, smart home networking, smart ware and traffic information system. In the U.S., relevant departments and research institutes cooperate with each other for ubiquitous computing research under the leadership of NITRD. In 2005 alone, which is the initial stage of the research, the U.S. injected USD 2.256 billion of immense budget, and has financially assisted the R&D of university research institutes and private sector companies through DARPA of the US Department of Defense, and NIST of the Department of Commerce [4]. In Europe, comprehensive ubiquitous IT policy is implemented by establishing the "i2010" strategy at EU level. To back up such a strategy, the budget for the 7th intensive R&D sector (2007~2013) was slated at 72.7 billion euros. Through this, we can see that investment increase for R&D is carried out in large scale [4].

As such, many people feel the need for ubiquitous computing environment, and strive for research activities. But, there are many obstacles in building ubiquitous computing environment in daily life. To build ubiquitous computing environment, the software that can be accessible to and can use each equipment used for the service is needed, and a lot of time and specialized manpower is required, which increases the relevant cost [2]. Also, it is difficult to change the already built ubiquitous environment structure or add new equipment to it. For example, a user can turn on/off an air conditioner outside of his/her home through Smart Wall Pad using Samsung SNS smart home solution. However, control of another company's air conditioner is impossible. To solve such a problem, it is necessary to standardize access to various equipment and control mechanism.

This paper proposes Web of Things (WoT) platform to build ubiquitous computing environment. The proposed technology enables to build ubiquitous computing environment with low cost by standardizing access to various equipment and control mechanism, centered on the WoT platform.

The remainder of the paper is organized as follows.

Chapter 2 reviews existing related works. Chapter 3 explains the Web of Things and REST architectural style. Chapter 4 proposes technology related to WoT platform model and Chapter 5 describes WoT platform embodiment. Chapter 6 shows the performance evaluation result. Chapter 7 summarizes and concludes this paper.

## 2. Related Work

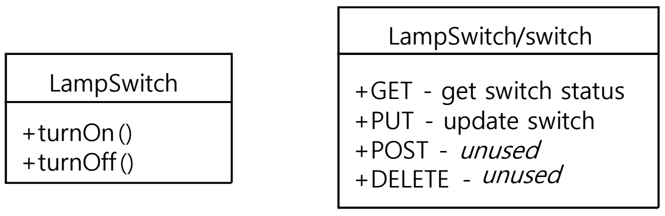
In April 2010, SG13 carrying out ITU-T's Next Generation Network (NGN) standardization initiated the development of standard recommendation (Y.WoT), which enables to supply real world things or services as one service entity through the Web [3]. In Europe, ITEA (Information Technology European Advancement) launched a SIREN (Service Infrastructure for Real time Embedded Networked Application) project in 2003 and conducted the project until 2005, aiming at developing service infra-structure for real time embedded network application. Following the project, ITEA will undertake a SODA (Service Oriented Device and Delivery Architecture) project, which is developing the perfect ecological environment, based on device-linked framework developed by the SIRENA project [1, 15]. In [5], the basics of the Web of Things architecture, based on RESTful principles and a smart gateway, are described. It provides a practical framework for users to build their own WoT devices and applications. Meanwhile, many researches integrating sensor systems using the wireless Internet technology can be found [11, 12, 13]. These researches propose the platform that can collect sensor information using Web service technology.

The WoT broker proposed in Y.WoT, and the smart gateway proposed in [5] are focused on offering the functions of things to users, but specific methods on the linkage of each thing with smart gateway or broker are not presented. SODA and sensor network researches [11, 12, 13] links the functions of smart things with the Web environment using Service Oriented Architecture. Unlike the REST technique, it is difficult to actualize an automatic integration function, due to no unification of interface.

## 3. Web of Things and REST Architectural Style

The Web of Things mean that they are expressed with resources to which things are accessible on the Web with the physical things integrated with the Web, and that the functions of things are built in an accessible service form.

The Web of Things can easily control the information and functions of things by using basic Web communications protocol (HTTP) and languages (HTML, JavaScript, XML, etc.). Namely, the cost required to build ubiquitous environment can be reduced. Especially, each thing's data and functions can be consistently identified and accessed on the Web by applying the REST technique-using Resource Oriented Architecture in embodying the Web of Things [1, 6, 8, 9]. Figure 1 compares the SOAP (Simple Thing Access Protocol) style interface used much as the existing Web service technology with REST architectural style interface in calling Smart Lamp object's function.



**(a) SOAP service style** **(b) REST architectural style**

**Figure 1. SOAP vs. REST architectural style**

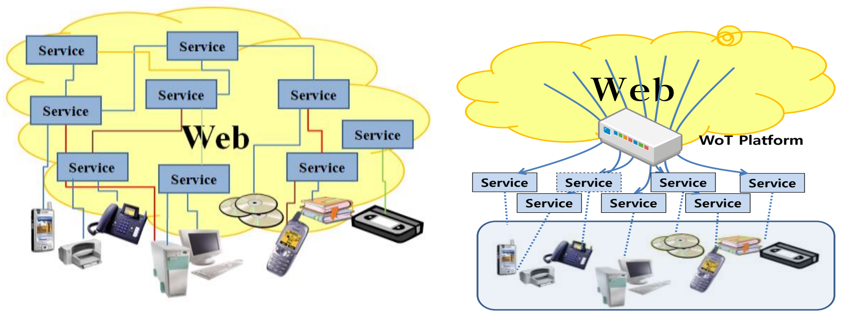
## 4. WoT Platform

### 4.1. WoT Platform Model

To build the Web of Things, WoT platform is necessary in order to manage various smart things and arbitrate functions. As demonstrated in Figure 2, WoT platform manages the information of smart things, and provides passage to receive smart things' services. Smart things refer to the devices that can communicate with WoT platform and dynamically provide services.

The functions of WoT platform are presented below:

- Management function: Registers and manages smart things as Web resource.
- Interface function: Provides the functions of smart things to the outside in the Web screen and Web service form.
- Security function: Offers a security function on access to smart things.
- Link function: Adjusts the data and functions between smart things to be linked mutually.



**(a) Web of Things Concepts** **(b) WoT Platform Model**

**Figure 2. Comparison of the general Web of Things and WoT platform model**

### 4.2. Registration Protocol

Registration protocol is used to register smart things on the WoT platform. Figure 3 exhibits registration procedure using protocol and the service use procedure of smart things.

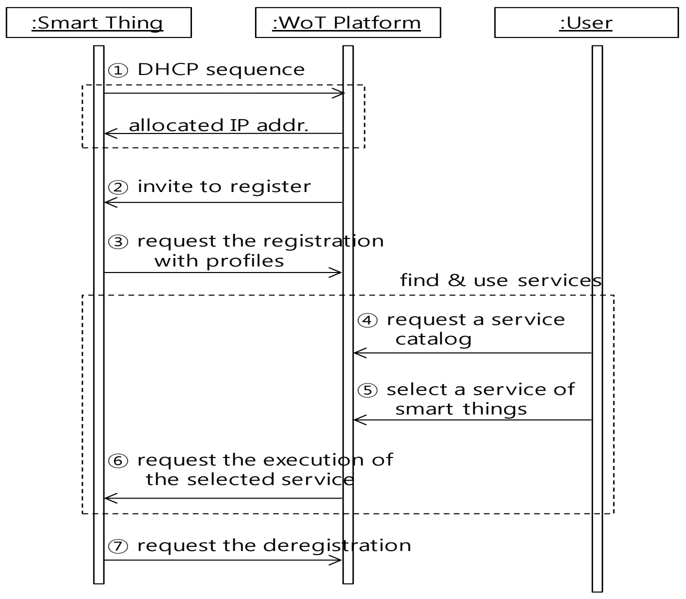

:Smart Thing
:WoT Platform
:User
① DHCP sequence
allocated IP addr.
② invite to register
③ request the registration with profiles
find & use services
④ request a service catalog
⑤ select a service of smart things
⑥ request the execution of the selected service
⑦ request the deregistration

**Figure 3. Registration & Use sequence**

① : IP address is allocated to a smart thing through DHCP to connect to the Internet. In doing so, WoT platform detects new smart thing's connection by including DHCP server function or analyzing DHCP packet.

② : WoT platform sends a message inviting registration to the newly connected smart thing.

③ : The smart thing that received a message to invite registration requests registration by sending the service profile describing its own functional services to WoT platform.

④ ~⑥: A user selects his/her desired smart thing among the registered smart things by connecting to WoT platform. To call the selected service function, the WoT platform requests that the service be performed to the smart thing concerned.

⑦ : Smart things can withdraw registration by sending a message to withdraw registration, if they want to.

## 4.3. Service Profile

Service profile contains information related to smart things' functions. Each function of smart things is expressed in a resource (or property) form, according to REST principles, and its operation includes POST (Create), GET (Retrieve), PUT (Update) and DELETE (Delete).

To express such information, this study develops SPDL (Service Profile Description Language) using XML.

Figure 4 is an example of describing the service profile of 'Lamp Switch' smart thing with SPDL. Here, there is a switch as property of the smart thing concerned, and this property can have on/off value. Figure 4 describes that the operations, which can access the property, are R (retrieve) and U (update).

```
<?xml verison="1.0" encoding="utf-8" ?>
<smartThing name="Lamp Switch">
  <properties>
    <property name="switch">
      <propertyType> on_off_type </propertyType>
      <access> R U </access>
      <accessURI> /switch </accessURI>
    </property>
  </properties>
</smartThing>
```

**Figure 4. A service profile example of smart lamp**

### 4.4. UI Code Generation & Execution

As explained above, each service of smart things is expressed as property in the profile and each property has a value. The value that each property can have is decided by the type of the property concerned. The property type can be basically a built-in type (int, long, string, boolean, dateTime, etc.) defined in XML Schema, or a user-defined type defined with <restriction>, <list> and <union>[16]. For example, the *on_off_type* used in Figure 4 can be defined as a user-defined type using <restriction> as shown in Figure 5.

```
<simpleType name="on_off_type">
        <restriction base="string">
             <enumeration value="on" />
             <enumeration value="off" />
        </restriction>
</simpleType>
```

**Figure 5. Examples of on_off_type declaration**

If a user-defined type is used for the service profile of a smart thing, the SML Schema file including the type definition is also transmitted to WoT platform like the service profile file in the smart thing registration process.

Table 1 demonstrates the examples of property, data type and available operations on each product, if home appliances used in everyday life are developed as smart things.

**Table 1. Property examples on smart home appliances**

| Name | Property | Data type | Values | Operations |
|---|---|---|---|---|
| Lamp Switch | switch | on_off_type | on / off | R(etrieve) U(pdate) |
| Heating controller | switch | on_off_type | on / off | R U |
| | currentTemperature | int | | R |
| | presetTemperature | int | | R U |
| Ventilator | switch | on_off_type | on / off | R U |
| | fanSpeed | pan_speed_type | high/ medium/ low | R U |
| Electric rice cooker | control | rice_cooker_control_type | keepingWarm/ cooking/ reheating/ suspension/ reservation | R U |

| | | | | |
|---|---|---|---|---|
| | status | rice_cooker_status_type | keepingWarm/ currentlyCooking/ suspension | R |
| | mode | rice_cooker_mode_type | whiteRice/ mixedGrains/ highSpeedCooking | R U |
| | reservationTime | dateTime | | R U |

WoT platform automatically creates user interface through smart things' profiles so that users can use smart things services by accessing the Web. A user can search smart things through the automatically created Web screen, and can execute desired function of the smart thing, after selecting it.

A user interface code is automatically created, according to the property type and operation type. The C (reate) and D (elete) operations are generally used, when elements are added to or deleted from set property. WoT platform creates JavaScript and HTML code, to which information on a new element is inputted or which enumerates registered elements and deletes them on the Web screen. R (etrieve) operation is an operation obtaining property information. When a user selects the property concerned, it is displayed on the Web screen, after status information is received from a smart thing using the "GET" message. As for U (pdate) operation, user input screen is decided, according to property type. For example, in the case of *on_off_type*, the *on* and *off* values are enumerated, and a user selects between those two, or switch shape is shown. And, interface code can be created in the form of toggle.

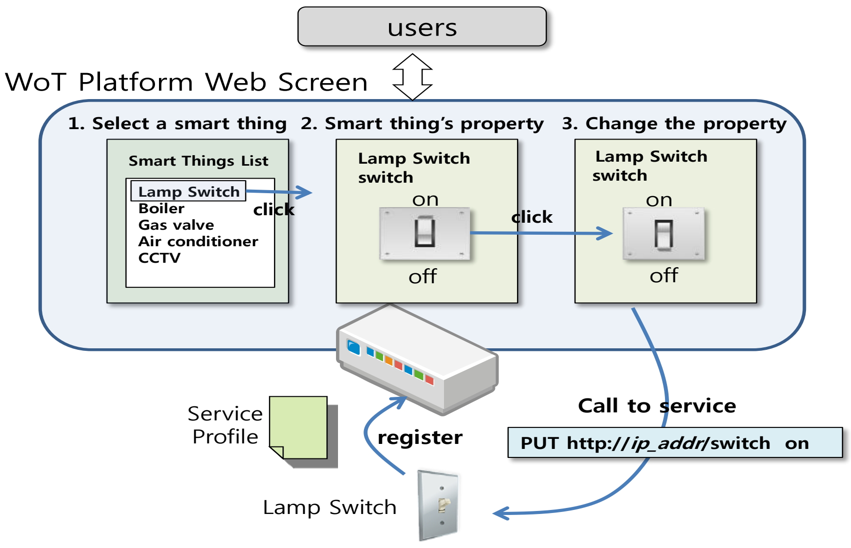


**Figure 6. Automatic creation and execution of Web screen for service access**

For example, if 'Lamp Switch' profile is registered on the WoT platform, the WoT platform shows the 'Lamp Switch' in the 'Smart Things List' on the Web screen as demonstrated in Figure 6 (If there are several lamp switches, a serial name follows after the name). If a user selects the item of 'Lamp Switch', there is a switch as its property, and the screen that shows values of the switch is displayed. According to user's selection, the user can turn on/off the switch.

The HTTP request message sending an order to change the value of switch, which is the property of ‘Lamp Switch’, to on (turning on switch) is shown in Figure 7.

```
PUT  /switch  HTTP/1.1
some http headers ...
<?xml version="1.0" ?>
<value> on  </value>
```

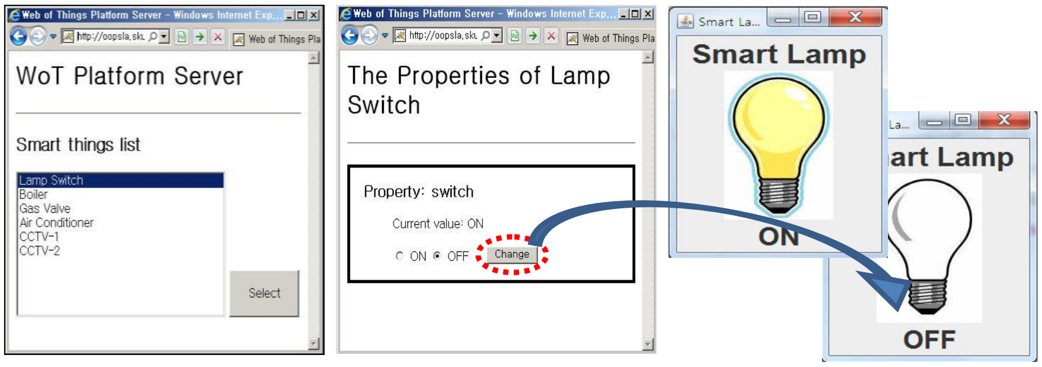
**Figure 7. HTTP request turning on Lamp Switch**

## 5. Prototyping WoT Platform

This research develops the proposed WoT platform’s prototype. The WoT platform is embodied with Java application and Tomcat Server 7.0 JSP, working in the Windows Server 2003 environment. To show that the system works, the ‘Lamp Switch’ program playing a role of a smart thing is developed in this research.

DHCP server is installed together in the server, where WoT platform is installed. Whenever a new smart thing is connected, the WoT platform checks it and conducts the registration procedure of the smart thing.

A user can access the registered function of the ‘Lamp Switch’ smart thing by accessing the Web interface of WoT platform. Every time an on/off order is made, the Lamp on/off image is displayed on the screen in the ‘Lamp Switch’ program.



WoT Platform Server

Smart things list

Lamp Switch
Boiler
Gas Valve
Air Conditioner
CCTV-1
CCTV-2

The Properties of Lamp Switch

Property: switch

Current value: ON

ON [✓ OFF]

**(a) Smart things list** **(b) Property list** **(c) Smart thing ‘Lamp Switch’**

**Figure 8. Lamp Switch control through WoT platform Web screen**

As exhibited in Figure 8, the WoT platform enables a user to access smart things’ function through Web screen and simultaneously provides RESTful API to support M2M (Machine to Machine). External programs can easily access the functions of smart things through the RESTful API.

For an instance, the platform-registered ‘Lamp Switch’ can be turned on/off through transmission of the messages demonstrated in Figure 9 by an external program.

```
PUT  /SmartThings/Lamp%20Switch/switch  HTTP/1.1
some http headers ...
<?xml version="1.0" ?>
<value> on  </value>
```

**Figure 9. Request messages sent to WoT platform to turn on Lamp Switch**

If ubiquitous environment is generalized, numerous objects used in everyday life can become smart things. To manage these, a directory service to register and inquire the addresses of smart things is basically needed. The WoT platform embodied in this study offers the directory service on smart things.

## 6. Evaluation

There are two methods for an external program to access smart things. One method is to directly access smart things, after identifying smart things' addresses with the directory service function. The other method is to indirectly access smart things through WoT platform.

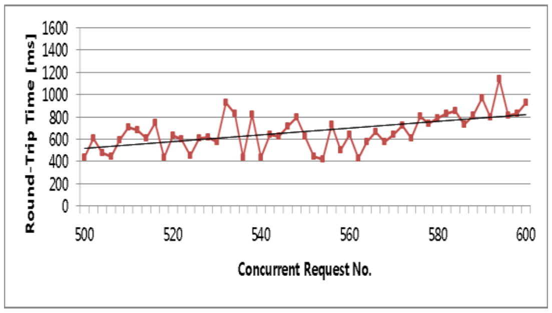
The indirect access method through WoT platform is more excellent in flexibility and security aspects than the direct method to access smart things, and the former is not behind in the performance aspect, compared to the latter. To support this statement, I have implemented a simple scenario where a user issues a GET request to read the current value of a smart thing, 'Lamp Switch.' This study measured the average round-trip time for each request, according to the two access methods mentioned above and the results are shown in Figure 10.

In the first case, status information is read by directly accessing a smart thing, after obtaining smart thing's address using a directory service function of WoT platform from a remote terminal. Measuring is conducted by gradually increasing the number of concurrent requests from 500 to 600. In this case, the average round-trip time is 667 milliseconds.

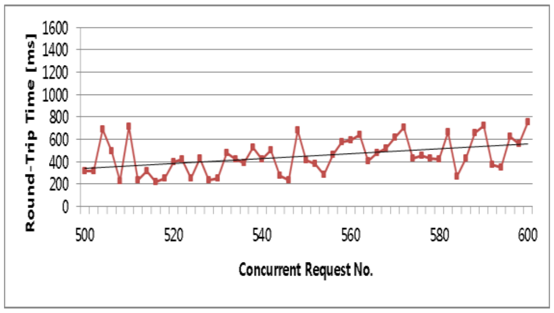
In the second case, the status information of a smart thing is requested to WoT platform from a remote terminal, and the WoT platform accesses the smart thing concerned, according to the request, reads the status information, and then, transmits it to the remote terminal. In this case, the average round-trip time is 449 milliseconds.

The reason why the indirect access method using WoT platform shows better performance result than the direct method to access a smart thing is judged to be that the WoT platform and smart thing exist in the same LAN, while the remote terminal exits in the remote distance from those two.

From this result, it is found out that a mode, in which the WoT platform manages smart things and an external user (or program) accesses the functions of smart things through WoT platform, cannot be a problem at all from the performance aspect.



(a) Round-trip time for direct access method



(b) Round-trip time for Indirect access method using WoT platform

**Figure 10. Performance comparison of a direct access method and an indirect access method using WoT platform**

## 7. Conclusions

This paper proposes the technology related to WoT platform for building ubiquitous computing environment. This paper designs communication protocol through which smart things can be registered on the WoT platform, and also develops a specification language that can draw up service profile describing the functions and access path of smart things. This paper also demonstrates that WoT platform can automatically create user interface code, through which a user can access the services of smart things, using service profile.

The paper presents contributions as follows.

a) Standardization of the access and control of various smart things

b) Demonstrating that the user interface of smart things can be automatically created.

c) In conclusion, the WoT platform can reduce time and cost in building ubiquitous computing environment and makes expandability possible. In addition, interoperability between devices can be enhanced using the WoT platform.

As a further research task, offering new services through combination of smart things registered on the WoT platform needs to be conducted.

## Acknowledgements

This Research was supported by SeoKyeong University in 2012.

## References

[1] Y. M. Park, A. K. Moon, H. K. Yoo, Y. C. Jung and S. K. Kim, "SOAP-based Web Services vs. RESTful Web Services", Electronics and Telecommunications Trends, vol. 25, no. 2, ETRI, **(2010).**

[2] M. In, K. Lee and S. Lee, "Web of Things (WoT) Standardization", TTA Journal, vol.138, ETRI, **(2011).**

[3] M. In, K. Lee and S. Lee, "Draft Recommendation Y.2063 - Framework of Web of Things - for consent", Study Group 13, TD 316, ITU-T, **(2012).**

[4] H. J. Choi, S. J. Yeon and W. G. Ha, "An Analysis on Ubiquitous IT of the United States and EU", Electronics and Telecommunications Trends, vol. 21, no. 2, ETRI, **(2006).**

[5] D. Guinard, V. Trifa and E. Wilde, "A Resource Oriented Architecture for the Web of Things", Proc. of IoT **(2010)** (IEEE International Conference on the Internet of Things), Tokyo, Japan

[6] D. Guinard, I. Ion and S. Mayer, "In Search of an Internet of Things Service Architecture: REST or WS-* A Developers' Perspective", Proceedings of Mobiquitous, **(2011)** (8th International ICST Conference on Mobile and Ubiquitous Systems), pp. 326-337, Copenhagen, Denmark.

[7] B. Ostermaier, M. Kovatsch and S. Santini, "Connecting Things to the Web using Programmable Low-power WiFi Modules", Proceedings of the 2nd International Workshop on the Web of Things, **(2011)**, San Francisco, CA, USA.

[8] R. T. Fielding, "Architectural styles and the design of network-based software architectures", PhD Thesis, University of California, Irvine, **(2000)**.

[9] R. T. Fielding and R. N. Taylor, "Principled Design of the Modern Web Architecture", ACM Transactions on Internet Technology, vol. 2, no. 2, **(2002)** May, pp. 115–150.

[10] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra and M. Spasojevic, "People, places, things: web presence for the real world", Mobile Networks and Applications, vol. 7, no. 5, **(2002)**, pp. 365–376.

[11] P. Schramm, E. Naroska, P. Resch, J. Platte, H. Linde, G. Stromberg and T. Sturm, "A service gateway for networked sensor systems", Pervasive Computing, IEEE, vol. 3, no. 1, **(2004)**, pp. 66–74.

[12] N. B. Priyantha, A. Kansal, M. Goraczko and F. Zhao, "Tiny web services: design and implementation of interoperable and evolvable sensor networks", in In Proc. of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys), New York, NY, USA: ACM, **(2008)**.

[13] A. Kansal, S. Nath, J. Liu and F. Zhao, "SenseWeb: an infrastructure for shared sensing", IEEE Multimedia, vol. 14, no. 4, **(2007)**, pp. 8–13.

[14] M. Weiser, "The Computer for the Twenty-First Century", Scientific American, **(1991)**, pp. 94-100.

[15] S. de Deugd, R. Carroll, K. Kelly, B. Millett and J. Ricker, "SODA: Service Oriented Device Architecture", IEEE Pervasive Computing, **(2006)** September.
[16] W3C, XML Schema Part 2: Datatypes Second Edition; W3C Recommendation, **(2004)** October 28, http://www.w3.org/TR/xmlschema-2/.
[17] W. C. Shin, "Building Ubiquitous Computing Environment Using the Web of Things Platform, Advanced Science and Technology Letters", vol. 43, Multimedia, **(2013)**, pp. 105-109, Science & Engineering Research Support society.

## Author

**Woo-Chang Shin**

He received the Ph.D. degree in computer engineering from Seoul National University in 2003, Republic of Korea. Currently he is a Professor at Department of Computer Science, SeoKyeong University His research interests include software development methodology, software modeling, software testing, and formal specification.