A Smart Object centric Indoor Location Model for Future Ubiquitous and Grid Services

Fahim Kawsar[1], Kaori Fujinami[2], Jong Hyuk Park[3], Tatsuo Nakajima[1] , Jeong Bae Lee[4], Kee Wook Rim[4]*
[1]Waseda Univerisy, Tokyo, Japan
[2] Tokyo University of Agriculture and Technology, Tokyo, Japan
[3] Kyungnam University, Korea
[4] Sunmoon University, Korea

Abstract

One of the primary requirements for future ubiquitous environment is to exploit location information to provide contextual services. Typical indoor location systems are heavily infra-structured making it difficult to deploy practically in home settings. In this article we propose a lightweight location model utilizing existing physical objects in our environment. Our approach is to use sensor augmented daily life objects surrounding us for extracting location information. Some of these everyday objects are static in nature and have designated location, like a bed in the bedroom, a refrigerator in the kitchen, etc. Utilizing this characteristics, we present "Spreha", a light weight hierarchical location model where static physical objects are used as reference points for identifying mobile objects like a chair, a watch, a lamp, etc. The model is organized in a tree structure representing the containment relationship and is independent of underlying sensing infrastructure. A prototype implementation of the model has been constructed as a pluggable module of a generic middleware using Bluetooth technology. This paper discusses about the design, architecture and findings of the prototype implementation. The implication of such lightweight location model is very important in realizing future environments where ubiquitous services will merge with grid services

Keywords: Pervasive Environment, Smart Object, Location Model

1 Introduction

Location is considered as the primary context for proactive service provision and has been investigated in various location-based services. This is particularly important for future ubiquitous environment where various pervasive services will exist to support end users. The merging of grid computing services with pervasive ones for better just-in-time service provision only makes the location information vital. Till date several location systems are investigated in the literature [4,5,12,14,16,18]. However, usually location based applications atop this systems are tightly coupled with the underlying location infrastructures that depend on particular sensing systems. In outdoor environment GPS has turned out to be useful for location based services, however assuming similar congested sensing infrastructure for indoor positioning is impractical and expensive. Although till date several researchers have explored indoor positioning, the outcome is insignificant. This is because of aiming at expensive sensing infrastructure and centralized database approach. Furthermore, due to inherent dependency on explicit senor most of these works assume that location privacy is the responsibility of the applications thus exhibiting no privacy protection at the infrastructure end. We argue that those approaches are not suitable for location model for indoor pervasive environment and henceforth we propose Spreha that provides an indoor location model exploiting the artefacts that are around us already.

We augment daily life artefacts like a chair, a table, a door, a mirror, a bed etc. with various kinds of sensors to capture contextual information and to provide ambient services. We call them Smart Objects [10]. Our vision is to utilize these objects (and their virtual counterparts) for value added proactive services in addition to their primary roles. Among these artefacts, many are static in nature, have designated location and we rarely move them, for example a couch in the living room, a cooking oven in the kitchen, a room door/window etc. Furthermore, the geographic containers of these artefacts are static. For example, a kitchen in the apartment 111 of Cherry Crest Building with address 1-21-2 Shinjuku. Tokyo, Japan. We exploited this static nature of these artefacts and their containers to construct the model by using

**Corresponding Author: Jong Hyuk Park*
Email: parkjonghyuk1@hotmail.com

them as location reference points and organizing them in a tree structure. When these artefacts are augmented with location sensing capability, they can identify the peer mobile artefacts within their vicinity. As a result we do not need any dedicated infrastructure as Smart Objects have their primary roles and location identification is just their value added functionality. Another interesting observation is the temporal requirement of location information meaning that a location model is only required when services are supposed to be served. Thus we do not need always-on location model using a central repository; on the contrary we can construct it only when it is required using the location providers around the service area at context. Therefore, by using Smart Objects we eliminate the requirement of centralized server, dedicated infrastructure and tight coupling with the underlying sensors. We believe this approach is feasible, practical and economical in pervasive environment. This is further important for merging grid services in ubiquitous environment. Typically, grid services are not designed to support location based services. Thus such weight system is very practically and suitable as it is very easy (as we will explain in rest of the article) to associate location property to an existing service using the proposed mode.

The rest of the article is organized as follows: section 2 discusses about the design guidelines. In section 3 we present the conceptual model. Section 4 discusses the prototype implementation. In section 5 a case study of Bluetooth (used as the sensing technology for the prototype) is presented with an illustrative example scenario. In section 6 we focus on some generic issues and in section 7 we briefly review the related work. Finally section 8 concludes the paper.

2 Design Guideline

Considering the nature of the proactive applications in pervasive environment, we have designed our proposed model following the requirement guidelines mentioned below.

No Dedicated Infrastructure: Location model should not be tightly coupled with underlying sensing infrastructure. Instead we should be able to extract location information in an adhoc manner from sparsely distributed location providers. Our model is based on Smart Objects that have primary roles and are already deployed in the environment in a distributed manner. We augment them to extract location information to construct the model thus eliminating the need of dedicated infrastructure.

Sensor Independence: Location sensing can be of two types: tracking and positioning. Tracking measures the location of other objects like RFID tag whereas positioning locates itself like GPS. Our model is designed to support tracking systems like RFID, Bluetooth, ultrasonic, infrared etc. To facilitate this support, the model has been logically decomposed into 2 layers, where the top layer represents the unified location information to the applications and the bottom layer handles the underlying sensing techniques.

Containment Relationship: For a useful location model, containment information is very important, as it allows application to exploit the collocated services. We have designed the model in a hierarchical manner using a tree organization where parent-child-sibling pattern are used to represent the containment relationship.

Privacy: For location-based applications, privacy is very crucial. Unlike other works in the literature, we consider location protection is a core responsibility of the location providers. Thus, we have designed the model with an explicit protection support that allows the location providers to control the exposition of location information.

No Centralized Database: The location data should not be stored in a centralized server as it restricts mobility and is vulnerable to single point failure. Since, we have used Smart Objects as the primary location identifier component of our model, we eliminate any need of centralized database. This is because Smart Objects are self-contained and are usually organized in a disperse manner.

Following this guidelines we have designed Spreha. In the next section the logical model of Spreha is presented.

3 Logical Model of Spreha

Spreha is primarily based on two conceptual components: Smart Objects and artefact container as shown in figure 1. Both of these conceptual components have virtual counterparts that construct the location model in the virtual space.

Smart Object: These artefacts are our everyday objects surrounding us. We augment these artefacts with sensing and computing facilities to provide

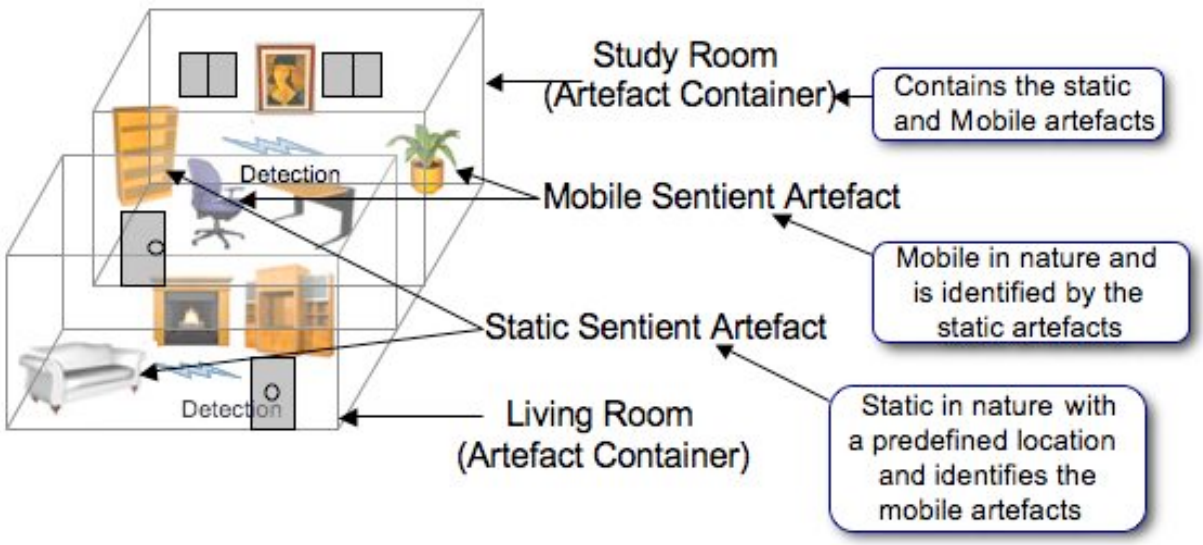
Study Room
(Artefact Container)
Contains the static and Mobile artefacts
Detection
Mobile Sentient Artefact
Mobile in nature and is identified by the static artefacts
Static Sentient Artefact
Detection
Living Room
(Artefact Container)
Static in nature with a predefined location and identifies the mobile artefacts

Figure 1: Conceptual Components of Spreha Model

value added services in addition of their primary roles. We have employed a profile-based approach to represent these real world physical artefacts in the virtual world [9]. A profile simply indicates the additional functionality an artefact can perform and usually a single artefact can provide multiple profiles. In Spreha, these artefacts are classified into following two types based on their mobility and functionality roles:

Static Sentient Artefact: These are our everyday artefacts that are static in nature and have pre-designated location. For example: A bed in the bedroom, a refrigerator in the kitchen etc. These artefacts implement the location provider profile and are used to identify the location of peer mobile artefacts that are proximity wise nearby to that artefact.

Mobile Sentient Artefact: These are our everyday artefacts that are mobile in nature like: a coffee mug, a chair, a cooker etc. They implement the profiles for which they are augmented. But they do not implement any location specific profile. Static Smart Object identifies them and tags them with its own location.

Artefact Containers: These are the containers of the artefacts, for example: a room, a floor, a building etc. A container can host other containers.

3.1 Containment Relationship

We have employed a tree-based organization for representing the containment relationship of the artefacts. For this, every static Smart Objects and artefact containers of the real world are assigned a unique identifier with its predefined location like, Fahim's desk, Fahim's room, 2nd floor etc. Also for static artefacts and containers, their parent container name (where applicable) is predefined. These predefined location are then marked with a level number like: Building Name: n, Floor: n+1, Room: n+2, Artefact 1:n+3, Artefact 2:n+3, Artefact 3:n+3 etc. Here n represents the higher-level location information of the primary container in an indoor environment that is a building. This n can be generated using outdoor location models like Google Maps [19]. Such model gives us very high-level location information as Country, City, Area and Street/Avenue. We can augment these locations with our proposed level number as Country:0, City:0+1, Area:0+2, Street:0+3. Since a static Smart Object is marked with this level number, when it identifies a mobile artefact, it associates this level number with the identified artefacts. Thus when an application communicates with the Smart Objects to retrieve location information, the application can instantly construct the tree based containment (with parent-child-sibling relation) organization of the surroundings. Figure 2 exhibits this particular feature.

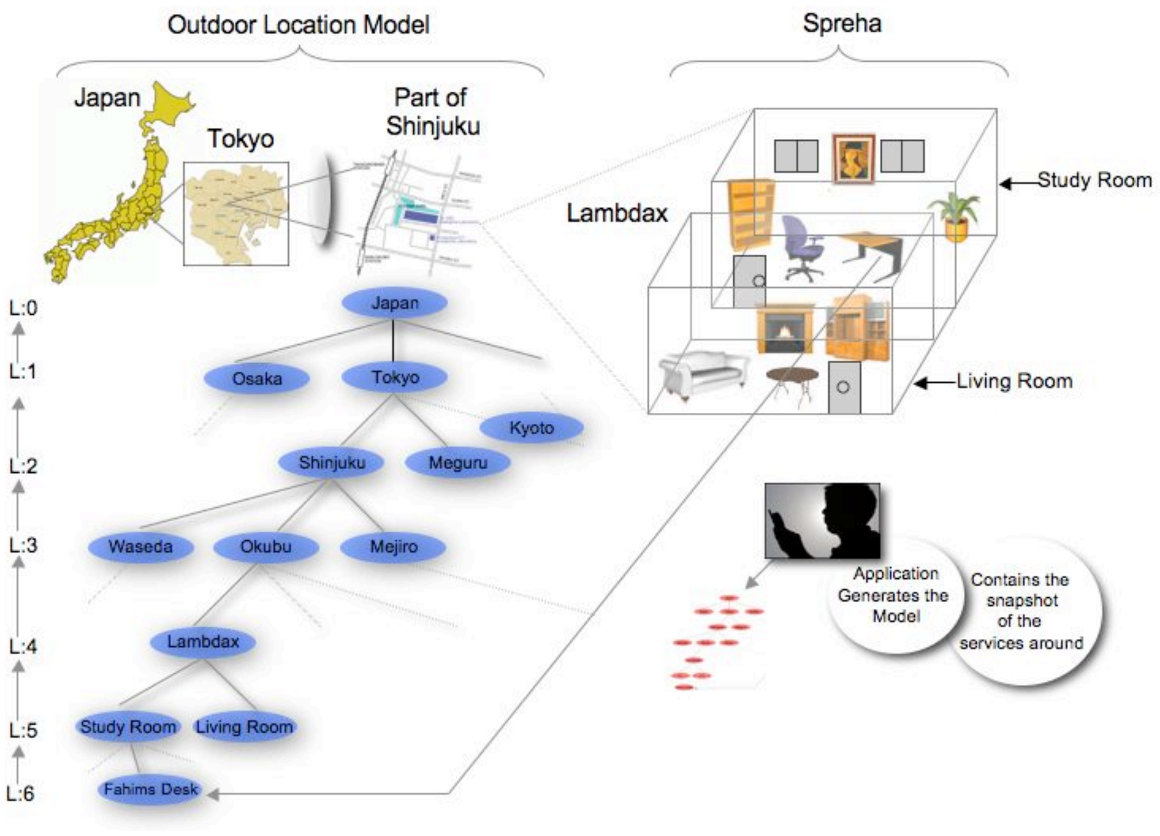

Figure 2: Containment Relationship in Spreha

3.2 Containment Relationship

In Spreha, static Smart Objects are the source of location information and the application communicates with them to retrieve location data. Since these artefacts are self-contained, it is possible to integrate a predefined privacy policy. Currently Spreha incorporates a rudimentary trust policy with static Smart Objects, which contains two attributes: public policy and private policy. Public policy means location information of the identified artefacts can be exposed publicly, whereas private policy means the opposite. Artefacts can provide their preferred policy during deployment time.

4 Prototype Implementation

We have implemented the logical model of Spreha as a pluggable module of a generic middleware Prottoy, designed to support the development of proactive applications integrating multiple Smart Objects [8]. For clarity, here we will briefly introduce Prottoy, which is composed of three core components. Those are:

- **Resource Manager**: Responsible for resource discovery, e.g. the Smart Objects.
- **Artefact Wrapper**: Responsible for encapsulating artefacts and offering artefact services and context information to applications.
- **Virtual Artefact**: Responsible for providing unified interface to applications for interacting with the underlying layers.

The details of these components are described in [8] and are out of scope of this paper. In this section we will only discuss about the artefact wrapper.

4.1 Artefact Wrapper as Spreha Core

The artefact wrapper component basically handles the Spreha specific tasks. This wrapper is used to create the virtual counter part of the Spreha's two conceptual components: Smart Object and artefact container. Developers are responsible to create this

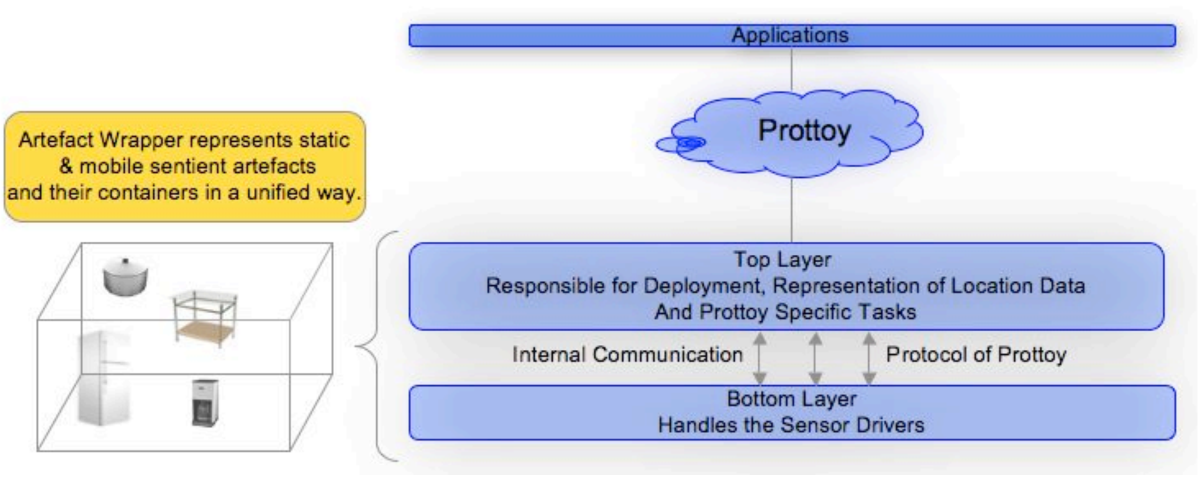
Applications
Prottoy
Artefact Wrapper represents static & mobile sentient artefacts and their containers in a unified way.
Top Layer
Responsible for Deployment, Representation of Location Data And Prottoy Specific Tasks
Internal Communication
Protocol of Prottoy
Bottom Layer
Handles the Sensor Drivers

Figure 3: The internal architecture of Artefact Wrapper

wrapper. Basically once the value added functionalities of an artefact is identified (service profile) and required sensors are augmented, the developers use this component to provide the sensor driver that implements the profile. So for static Smart Objects, they implement the *location identifier* profile and for artefact container they implement *container* profile. We mentioned in the guideline section that our design concern is to separate the profile specification from the sensor implementation. The internal design of the artefact wrapper satisfies this as shown in figure 3. Artefact wrapper's top layer has several other sub modules that takes care of other Prottoy related tasks and are out of the scope of this paper. But in general the top layer represents the overall functionality of the artefact that the developers can initialize during the deployment time and Prottoy handles the details.

The top layer of artefact wrapper is a ready to run component from Prottoy that the artefact developers use to specify initialization information. A GUI is provided to the developer to state this information during the deployment time of the artefacts. The followings fields are specific to Spreha that developers should provide: Static location (for static artefacts and containers), Parent Container Location, Location level (for static artefacts and containers), Privacy policy and Service profile like location provider, artefact container etc. In the current implementation, simple IP filtering technique has been used as the privacy policy of the artefacts. So, static artefact only responses to a request for location information that comes from allowed range of IP addresses. The bottom layer of the artefact wrapper provides unified API to developers to implement the underlying sensor specific codes. So, if the application environment depends on Bluetooth for sensing, the developers implement the Bluetooth driver in this layer. Similarly for RFID, WiFi or Infrared based environment, they simply need to modify this layer. In the section 4.2 we have presented the programming model that illustrates these processes.

4.2 Deployment and Interaction of Artefacts

The artefacts deployment process is depicted in Figure 4 where in line (1)-(4) the artefacts communicates with the resource manager to provide their profile, level number, static location and security policy. In (5) – (6) the artefacts start the discovery process to identify the nearby peer artefacts. For using artefacts and retrieving location information, application initially communicates with the Resource Manager, and once the identity of the artefacts are found from resource manager (as a consequence of artefacts wrappers deployment process), the application can directly communicate with the artefacts for retrieving location data in addition to artefacts other services if any. In the current version, two communication modes are provided: Polling and Subscription. Since most of the current tracking sensor technologies are radio based, static artefacts associate Detection Time/Time of Flight (TOF) and Received Signal Strength Indicator (RSSI) with the discovered artefacts, which are internally used by Prottoy to resolve conflict and increase identification accuracy.

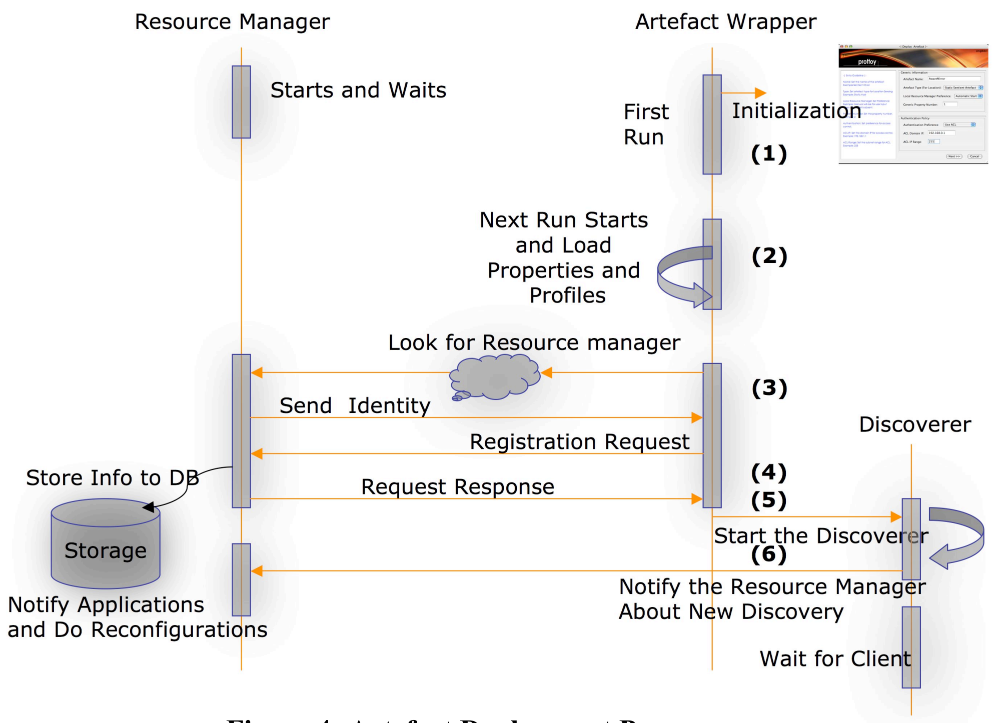

Figure 4: Artefact Deployment Process

That means when two static artefacts identify same mobile artefact in their vicinity, these two parameters are used in conjunction by Prottoy to tag the correct location. However, all these phases are encapsulated by Prottoy's virtual artefact component [8]. In the next subsection, we will present the programming model for the application developers to demonstrate these interactions.

4.3 Programming Model

Prottoy basically has two types of developers. One who develop the artefact wrappers (Smart Objects, containers) and the others who develop the proactive application using Prottoy and Smart Objects. For artefact developers Prottoy provides very simple APIs. We have mentioned artefact wrapper is designed based on profile notion [9]. So, artefact developers only add a profile and implement the profile handler. The following code fragment shows this process. Here, first developers add a location provider profile for a mirror that is in a washroom and then implement the handler for that profile. Similarly for container wrapper, developers implement the container profile which essentially just a location name. If the artefacts support multiple profiles based on functionalities, developers just need to add profiles and implement the handler for those functionalities. Once a static artefact is deployed, it periodically runs the embedded discovery service to locate nearby peers and stores the discovered peer information in a local cache until the next run. In the next discovery phase the previous location data is overwritten.

```
ArtefactWrapper wrapper = new ArtefactWrapper();
MirrorHandler handler = new MirrorHandler
                    ("Location-Provider",wrapper);
wrapper.addProfileHandler(handler);

public class MirrorHandler extends ProfileHandler{
public updateContext(){}
public locationCache(){}
public executeService(Service args){}
}
```

For application developers, Prottoy provides simple APIs that developers need to use through the virtual artefact component. Application can subscribe or query an artefact to get the location data and can generate the location model (the tree in this case) using Prottoy API. The following code segment shows these interactions.

```
VirtualArtefact mirror = new
      VirtualArtefact("Location-Provides","Washroom");
if(mirror.status){
mirror.addLocationListener(this,"callback");
}

QueryProcessor.getArtefactByLocation("Lambdax");

public void callback(locationData data){ }
```

Here, application developer creates a virtual artefact for the mirror at washroom and then subscribes to it for location data. The location data returned to the callback contains the artefact name, location, container location, level number, TOF and RSSI. Developers can call Prottoy utility to generate a location tree for deducing the containment information using this structure if necessary. Since all the interactions of the application with underlying artefacts are done through the virtual artefact, Prottoy can internally resolves the conflict among the artefacts location if any, by using the TOF and RSSI. For example, in the above code segment if we have multiple artefacts and if the application subscribes to both of them for location data, then Prottoy will use the TOF and RSSI internally to resolve the conflict if both the artefact claim about the presence of the same mobile artefact at their individual vicinities. Thus, the application callback will receive only one location for the artefact that is accurate according to the resolver of Prottoy. Developers can also use QueryProcessor to find the artefacts in a specific location. Prottoy internally communicates with static artefacts registered to the environment to generate the responses. In summary this mechanism can be enumerated as follows:

- Prottoy internally checks the resource manager to find the artefact that is in that location.

- If it's a container then Prottoy recursively check all the artefacts that have this container as parent.

- All the artefacts are then communicated to extract the location of the mobile artefacts in their vicinities.

- If the required location is not a container, then this method returns all the mobile artefacts seen by a static artefact at that location including itself.

However, in both subscription and query cases, applications' IP is checked by the location provider artefact with the range of allowed IP for confirming that the application is not a malicious one. In the current implementation, we do not have any integration with outdoor location model as mentioned in section 3. So, the top-level location is the name of the building. In the next section we will present a case study of our sensor implementation using Bluetooth technology.

5 A Case Study with Bluetooth

We have mentioned and shown in the previous section that Spreha is loosely coupled with underlying sensing infrastructure. Any suitable tracking sensors can be used with Smart Objects to provide the location data, as long as the artefacts are deployed using artefact wrapper. In our prototype implementation we have selected Bluetooth as underlying sensing technology. Bluetooth is a short range, low power open standard implementing wireless personal area network and it uses the unlicensed 2.4GHz ISM band exploiting frequency hopping scheme to avoid inference. In order to communicate, Bluetooth devices organize themselves into small networks called piconets that composed of one master node and up to 7 slave nodes, in which the frequency-hopping scheme is synchronized and controlled by the master [17].

We primarily selected Bluetooth over other prevailing technology due to the following reasons:

- Wide acceptability and availability of Bluetooth in information appliances.
- Inexpensive in cost.
- Minimal Inference with other Radio Frequencies like IEEE 801.11a etc.

In this section we will first present an application scenario followed by performance analysis of Bluetooth.

5.1 Dormitory Guide

Several applications have been developed on top Prottoy using Spreha, but due to space constrain, here we are presenting only one application that we named "Dormitory Guide" shown in figure 5. The application is rudimentary cyber guide type application that helps a user to get familiar with a dormitory rules and mates. An example scenario is as follows:

"Thomas just moved to Wakeijuku, a famous student dormitory at the heart of Tokyo. Upon his arrival the dorm leader gave him a small handheld computer and a headphone and asked him to move around the dormitory. As Thomas started moving from living room to kitchen to Room 111, 112,…., he heard a short literal description of the rules of using the common rooms and about the students living in different rooms."

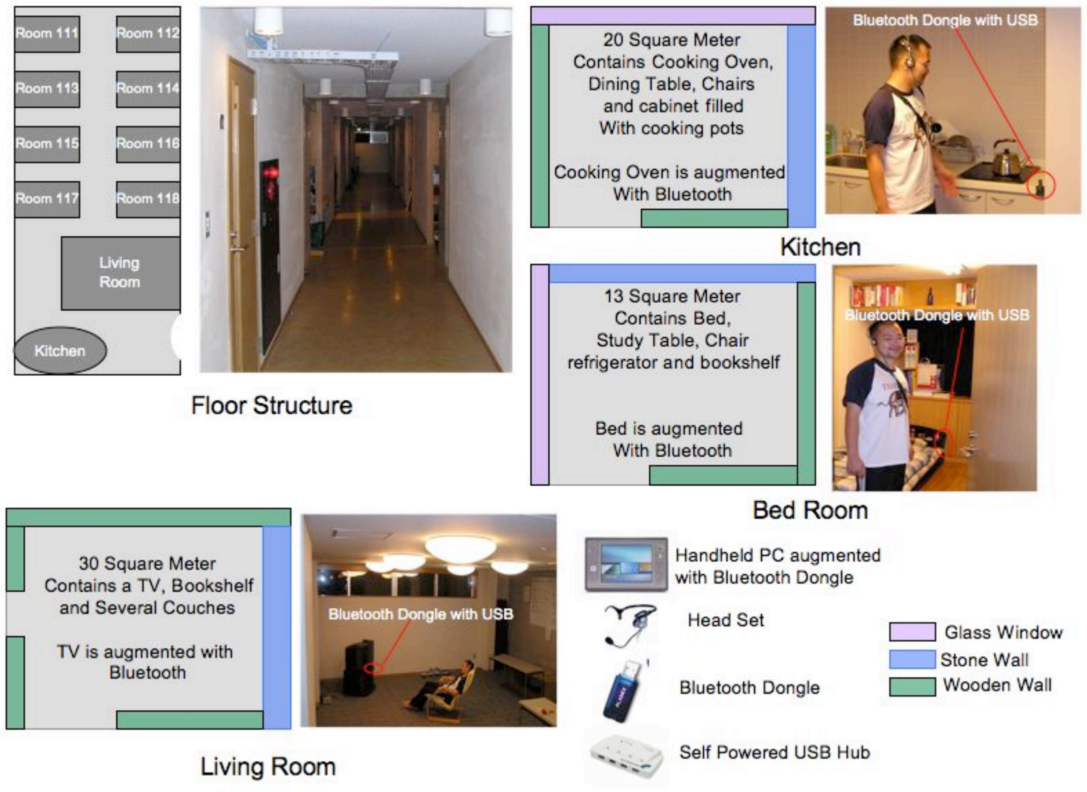

Figure 5: Dormitory Room Structure and Application Components

The application is very simple in functionality and in actually being installed and tested in the first floor of a dormitory at Tokyo, Japan. The structure of the first floor is shown in figure 7. Each student room is about 13sqm and the kitchen and living room is about 20sqm and 30sqm respectively. A Bluetooth chip is augmented in a static artefact at every room; bed in the student bed room; TV in the living room and the cooking oven in the kitchen. These artefacts were deployed using artefact wrapper using the room name as static location and Wakeijuku as container. The handheld is augmented with a USB interfaced Bluetooth dongle. The dongle is primarily initialized by a personal computer and then connected to self-powered USB hub. The complete unit USB hub + Bluetooth dongle is then installed in the designated static artefacts. Artefact wrapper wraps these artefacts. The handheld is also augmented with a Bluetooth dongle. The application runs in the handheld and it communicates with Prottoy to subscribe to the static Smart Objects for location data. During the movement when the artefacts in the room identify the handheld, application is notified and the associated description of the room is played on the handheld. In the next section findings about the performance are mentioned.

5.2 Bluetooth Performance

In terms of functionality, the Dormitory Guide application was successful primarily because of the uniform and sparse layout of the dormitory. However, during the testing phase, we have identified some interesting issues related to the location discovery process of Bluetooth; the detection speed with relation to RSSI. In this section we will briefly present our analysis result that was exhibited in our test and the resolution that we have reached.

The primary task of the locator in our model is to just to identify the artefacts in its vicinity. So, in case of Bluetooth technology, the locator just need to discover the near by artefacts. In Bluetooth, Piconet formation has two steps: the inquiry process where the master device discovers the nearby slave devices and secondly the paging process where connection between them are established for communication, like requesting the friendly name of the device etc.

5.2.1 Detection Speed (Time of Flight)

Due to the pseudo random selection of the sequence bits selected from 32 channels during inquiry process and the way a scanned device responses, we have seen some variety in the inquiry response time. Often we have received the query response instantly and often it took more than 10 seconds and often longer, used for playing the clip on the handheld when the person is in front of the same room in different times. Figure 6 shows the detection speed of the devices of 100 test incidents.

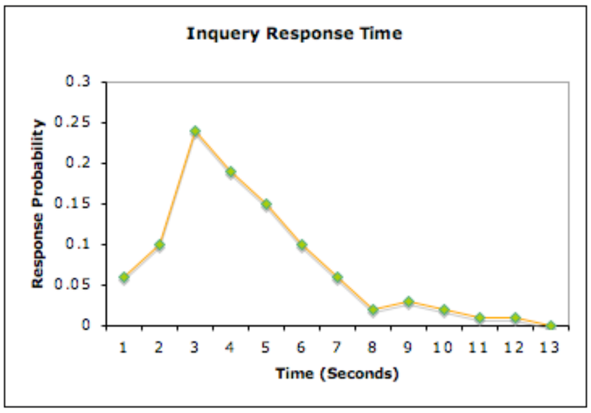

Figure 6: The Inquiry Response Time

The variation might be caused by the following reasons:

- Inference from other devices in ISM band range.
- A device listens for one of the 32 channels of Bluetooth at a time. Usually during an inquiry, a locator will inquire on half of those channels for 2.56 seconds and to other half for another 2.56 seconds and consequently two more times making in total 10.24 seconds for the discovery. So it may happen that locator will not even enquire on the same channel on which listener device is listening.
- Usually when a listener receives a request it goes to a back-off stage for 0 to 0.33 seconds randomly to minimize the collision.

The response of the inquiry is a 48-bit Bluetooth address. However if we need to retrieve the device name and other information we need to proceed to the next phase, which is paging; the most time consuming phase of the discovery process. Usually a timeout is associated with the paging process (Bluetooth Specification recommend 5.12 seconds as page timeout). We have tested the paging process with 20.48 seconds timeout and found that 83% time the paging was resolved within 5.12 seconds as shown in figure 7.

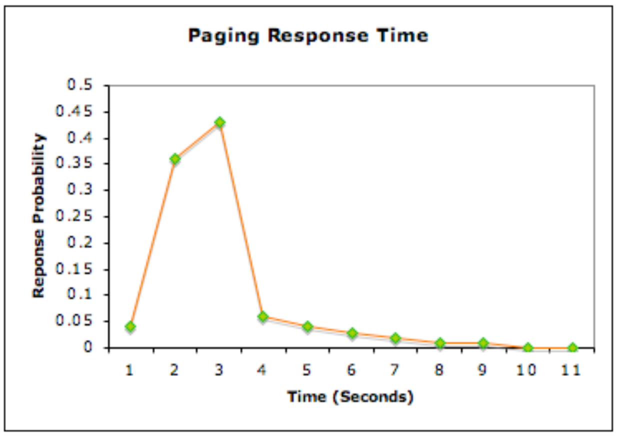

Figure 7: The Paging Response Time

The implications of these two analyses are that: we can adopt two schemes for associating the detection process for faster response:

Case A: We can maintain a local cache of the device name in the Smart Object (our locator) space that is used to tag the device name with address when identified. Thus we can eliminate the paging phase and can increase the detection speed; e.g. TOF.

Case B: Alternatively, we can use the both inquiry and paging process to retrieve artefacts data.

Although case A is faster, one problem is, when new artefacts are introduced, we need to update the cache of locators.

5.2.2 Signal Strength

Our next concern is the RSSI. Since, Bluetooth is a just a variation of radio signals, it is expected that the RSSI analysis depicts the similar characteristics like other works [2,12,14]. In our case, we are interested to see the relation between the RSSI and the distance. Although we have not used this relation for radio mapping like [2], we have combined this RSSI pattern with TOF pattern to resolve the conflict when, a same artefact is seen by two static artefacts in two different vicinity. Figure 8 shows the average signal strength measurement for three different distances. The data shown here is collected from calculating the RSSI received from an artefact at three differences in 100 test runs for each distance. It should be noted here that the tests were performed in the dormitory rooms where each room is equipped with some wooden furniture surrounded by cemented wall. Thus this result might not be same in other environment as 2.4 GHz frequency range is highly susceptible to occluding structures.

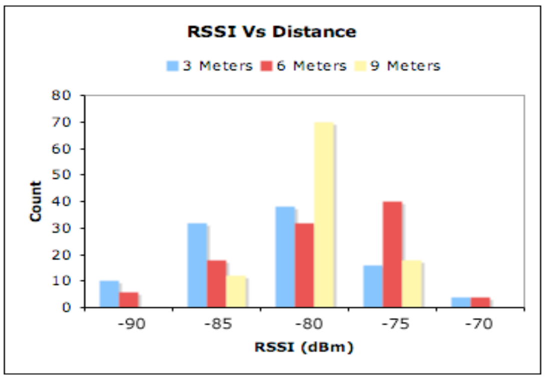

Figure 8: Avg. reading of Signal Strength at different distance

5.2.3 Combination of RSSI and TOF

The outcome of figure 6,7 and 8 depict the pattern of detection time and RSSI during the discovery phase of our system. One potential problem is that when the distance between two or more static artefacts are small (less than 10 meters), multiple artefacts can claim one mobile artefacts presence in its vicinity. Calculating only TOF to resolve this conflict might not lead us to the right prediction due to the varying pattern. For example: If an artefact detection time is longer that another artefact, not necessarily it implies that the first artefact is far from the locator due to the reason shown in figure 6 and 7. Similarly as figure 8 shows, we have received similar RSSI (for example: -85 dBm) from artefacts that are in varying distances.

To resolve this problem we have combined the RSSI and TOF using an optimistic algorithm that implies shorter TOF and better RSSI are the winner when there is a conflict. In spite of the simplicity of the algorithm we have found better result than using TOF only. In figure 9, we have demonstrated 3 cases for clarification of this issue, Two static artefacts are located by a difference of 8~12 meters and one mobile artefact is at varying distance from these two artefacts. We have selected this distance considering the 10-meter range specification (± 2). We have run the discovery service in both the static artefacts 100 times simultaneously and found the result depicted in the pie chart for each case with and without using the algorithm. An 8.6% improvement has been observed in overall accuracy when being used with the optimistic algorithm. It should be noted here the test was performed in an open space that means there are no obstacles between the artefacts. So, this result might not be directly applicable for all scenarios because of Bluetooth's susceptibility to occluding structures.

6 Discussion

In this section we will focus on some generic issues related to Spreha.

6.1 Design Features

Spreha is a lightweight location model for Smart Objects. So, the design principles of Spreha were to facilitate maximum support and flexibility for Smart Object based computing. Spreha is lightweight, independent of sensors and infrastructure, provides privacy support, exhibits containment relationship and manages location data in distributed manner. In this paper we have shown how Spreha satisfies each of this design features for supporting the optimum location based application integrating Smart Objects. However, we do not claim Spreha is a general location model applicable to all indoor applications. For example, right now Spreha has no feature for supporting mobile users. Also Spreha is limited to indoor facilities only and it is not applicable to outdoor environment in general. Spreha's primary concern is indoor applications that employ our Smart Object notion.

6.2 Smart Object Role and Cost

The primary advantage or contribution of Spreha is the utilization of Smart Objects as locators. By doing so we have eliminated the cost of dedicated infrastructure. Smart Objects are already available and have their primary roles. For example: a mirror in the washroom can act as an ambient display in addition to its primary role of a reflector. When we add location-sensing capability to it, we are just adding value to already existing artefacts, which has some basic roles to play. Thus our approach provides an economical way to location sensing. The basic cost of our system is a Bluetooth chip attached to an artefact, which typically cost US$25. If the artefact has an additional computing facility, like mirror as an ambient display, then the Bluetooth chip can be augmented with that. Otherwise, we just need to initialize the Bluetooth chip and then can attach it to the artefact using a self powered USB device. So, at extreme case for each reference point, the cost is Bluetooth dongle plus the USB adapter. In most cases, these can be integrated with other facilities of Smart Object thus reducing the cost. In comparison, other indoor location systems cost thousands to tens of thousands of US dollars for a

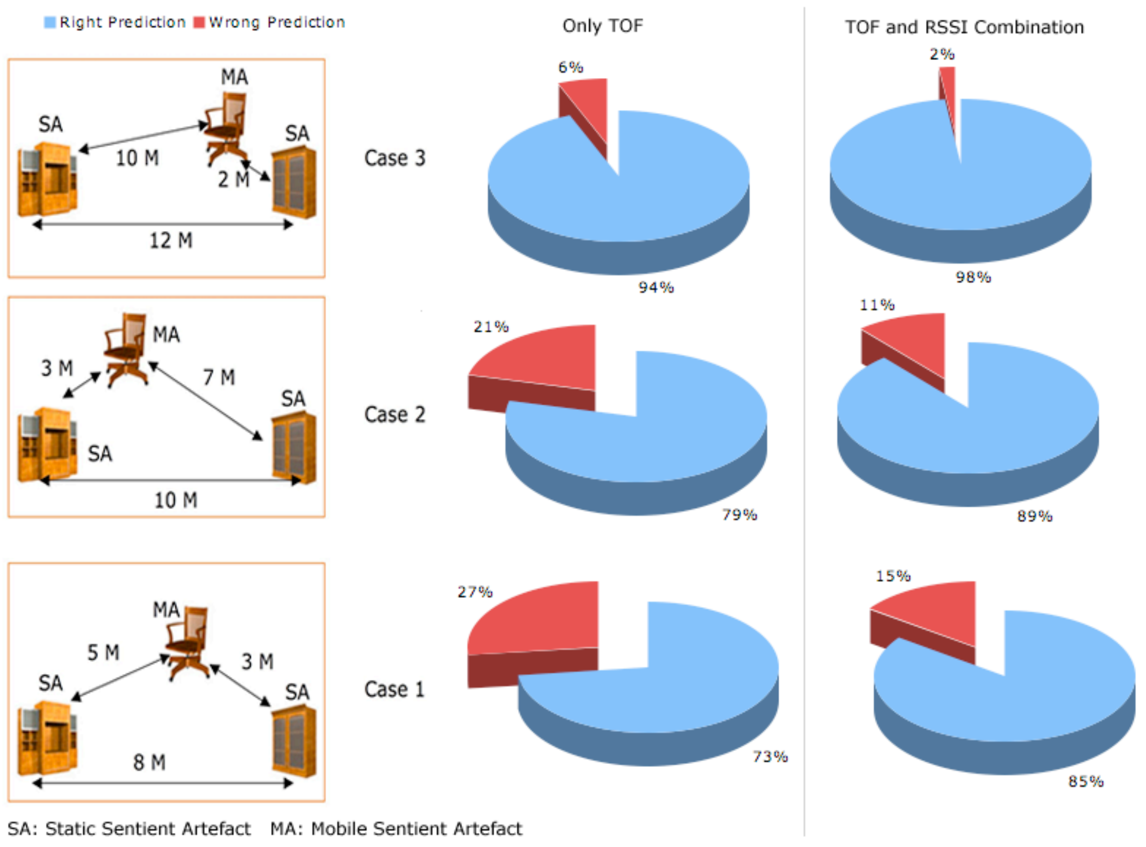

Figure 9: Prediction Performance of the locators.

1000 square meter installation for dependency on explicit sensor infrastructure.

6.3 Predefined Layout and Level Number

The outcome of figure 11 implies that we need a prior layout for deploying the reference points, e.g. which artefacts to use for location sensing and in what location. Also for the numbering scheme that Spreha uses for containment support requires pre assignment of level number to artefact and container. Although it is an overhead, we don't consider it as a shortcoming of our approach. Usually when we move into a new place, we put enough thoughts on putting the furniture in proper place that reflects our aesthetic and design sense. Considering this practice, we can assume that designing the Spreha layout is just an additional deployment task that users need to carry out to have the value added location aware proactive services. Furthermore, once the initial layout is done, we can easily add new artefacts in the infrastructure.

6.4 Accuracy

Because of the light weightiness of our approach, the accuracy is not very precise and roughly at few meters range. For some time critical systems this might not be acceptable. As we have mentioned Spreha is meant to be a location model for Smart Object based computing where the accuracy requirement is not as precise as real time systems, rather precision at few meters range is quite enough for contextual service provision.

6.5 Sensor Independence

Selection of the sensor for location estimation depends on application requirements like accuracy, operational range etc. In this paper, we have used Bluetooth as the underlying sensing technique. But this is not a strict requirement of Spreha. Bluetooth can be replaced with other technology like IEEE 802.11 wireless standards. For example: Ekahau tags and tag readers (http://www.ekakhau.com) can be used in Spreha without any major modification. Similarly WiFi access points have been used in many pervasive projects for location estimation. Spreha can be seamless integrated with those location-sensing systems. The strength of Spreha is not the sensing technology but the idea of using Smart Objects as a location reference point.

6.6 Tracking and Positioning

In current Spreha prototype, we have only supported tracking sensors. However one interesting extension might be supporting positioning by using RSSI and TOF signal analysis like trilateration. Considering Bluetooth is just another radio signal it is expected the result of this mechanism might exhibit similar insignificant behaviors as [2,12].

7 Related Work

In this section we will focus on some generic issues related to Spreha There are many researches and commercial efforts for providing location-based services. Most of them focuses on GPS based location data, like map viewers or navigation systems etc. However most of them lack any generalized location model. NEXUS [3] and HP Cool town [11] transforms geographic data collected from GPS to symbolic notation to identify the region that contains the objects. However, they don't support any tracking sensors and don't provide any privacy support. RAUM [4], ParcTab [15] and AURA [6] provide symbolic models as higher-level names with containment relationship and are independent of sensors like ours. However, their systems are based on centralized approach and privacy support is not at the infrastructure end. The primary distinction of Spreha with other indoor location system is an intellectual one because of the utilization of Smart Object instead of dedicated infrastructure. For example there are numerous indoor location systems that make use of ultrasonic [12,14], infrared [16], ultra-wideband radio [18], computer vision [5]. All these systems require a hardware infrastructure be installed in the environment and the systems are tightly coupled with the underlying sensors thus limiting the portability. Most importantly these systems are generally expensive, costing thousands to tens of thousands of US dollars for a 1000 square meter installation. These systems primarily focus on optimizing accuracy rather than wide-scale deployment. We consider these systems are not suitable for Smart Object based computing because of such inherent dependency on infrastructure. Place Lab proposes using existing GSM/RF/WiFi base stations as reference points thus eliminating the dedicated infrastructure issue [13]. Spreha is greatly inspired by their work but Spreha also introduces few features that are missing in Place Lab, like artefact end security policy, distribution of location information in static artefacts and a conceptual hierarchical location model. Using Bluetooth for indoor location sensing is not a new observation as it has already been explored in [1,7]. These works do not provide any location model and depend on centralized repository. Also, privacy provision is missing in their systems.

8 Conclusion

We presented a lightweight location model using the Smart Objects. Usually pervasive applications utilize various real world artefacts. Thus enabling some of those artefacts for location sensing provides a feasible approach as it eliminates the necessity of dedicated infrastructure and centralized repository. This is the primary advantage of Spreha. In addition our model represents the containment relationship with privacy support and is independent of sensor technology. Considering the common pattern of pervasive application, we believe our approach is economical and practical. We have also developed a prototype implementation with Bluetooth and reported the findings.

Spreha is very suitable for associating location property to existing grid services thus making compatible in a ubiquitous environment. We have shown through a prototype implementation that how Spreha can easily extend future services. We believe, the merging of grid services in ubiquitous environment will be very dynamic in nature and definitely need to exploit location information to provide just in time services. However, a heavily infra-structured environment poses significant engineering challenges to merge location information to grid services. In this article we have shown a very light weight location system, that can easily be merged with existing grid services to associate location information, due to simple and flexible model.

Finally we would like to work on further issues that are currently not available in our system like integrating outdoor model seamlessly with dynamics handover and supporting mobile users. Also the privacy support in Spreha is very primitive, and we are trying to incorporate more sophisticated techniques. Another extension that we are currently working on is the integration of spatial database technology for providing more flexibly query features. Furthermore, we would like to develop more realistic application on top our model. We hope soon we will be able to come up with some interesting results.

Acknowledgment

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) Support program supervised by the IITA(Institute of Information Technology Advancement)" (IITA-2009-C1090-0902-0020).

References

[1] Anastasi, G., Bandelloni, R., Conti, M., Delmastro, F., Gregori, E., and Mainetto, G. Experimenting an Indoor Bluetooth-Based Positioning Service. In *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03), 2003.*

[2] Bahl, P., and Padmanabhan, V. N. RADAR: An In-Building RF-Based User Location and Tracking System. In *Proceedings of IEEE Infocom* Israel 2000

[3] Bauer, M., Becker, C., and Rothermel, K. Location Models from the Perspective of Context-Aware Applications and Mobile Ad Hoc Networks. *Personal and Ubiquitous Computing*, vol. 6, Issue 5-6, pp. 322-328, Springer, 2002

[4] Beigl, M., Zimmer, T., and Decker, C. A location model for communicating and processing of context." *Personal and Ubiquitous Computing, 6(5–6): 341–357, 2002.*

[5] Brumitt, B. L., Meyers, B., Krumm, J., Kern, A. and Shafer ,S. EasyLiving: Technologies for Intelligent Environments. In *Proceedings of International Symposium on Handheld and Ubiquitous Computing*, pp. 12-27, 2000

[6] Garlan, D., Siewiorek, D., Smailagic, A. and Steenkiste, P. Project Aura: Towards Distraction-Free Pervasive Computing*, IEEE Pervasive Computing*, vol. 1, 2002

[7] Hallberg, J., Nilsson, M., and Synnes, K. Positioning with Bluetooth. In Proceedings of the 10th International Conference on Telecommunications (ICT 2003). 2003.

[8] Kawsar, F., Fujinami, K., and Nakajima, T. A Middleware for Sentient Environments", In Proceedings of The 2005 IFIP International Conference on Embedded And Ubiquitous Computing (EUC-05), 2005.

[9] Kawsar, F., Fujinami, K. and Nakajima, T. Exploiting Passive Advantages of Smart Objects. In *Proceedings of the 2006 International Symposium of Ubiquitous Computing Systems*.

[10] Kawsar, F., Fujinami, K. and Nakajima, T. Augmenting Everyday Life with Smart Objects, In Proceedings of the 2005 joint conference on Smart objects and ambient intelligence 2005, ACM Press, pp: 141 – 146.

[11] Kindberg T., et al, *People, Places, Things: Web Presence for the Real World*, Technical Report HPL-2000-16, Internet and Mobile Systems Laboratory, HP Laboratories, 2000.

[12] Priyantha, N. B., Chakaraborty, A., and Balakrishnan, H. "The Cricket Location-Support System" In *Proceedings of MOBICOM 2000.*

[13] Schilit, B., LaMarca, A., Borrirllo, G., Griswold, W., Mcdonald, D., Lazowska, E., Balachandran , A., Hong, J. and Iversion. V. Challenge: Ubiquitous Location-Aware Computing and the Place Lab Initiative. In *Proceedings of the First ACM International Workshop on Wireless Mobile Applications and Services on WLAN (WMASH),* 2003

[14] Want, R., Hopper, A., Falcao, V. and Gibbons, J. The Active Badge Location System., *ACM Transactions on Information Systems*, 10, 91-102. 1992

[15] Want R., Schilit, B., Norman, A., Gold, Goldberg, D, Petersen, K., Ellis, J., and Weiser, M. An Overview of the ParcTab Ubiquitous Computing Experiment, *IEEE Personal Communications*, Vol 2. No.6, pp 28-43, December 1995.

[16] Ward, A., Jones, A. and Hopper, A. New Location Technique for the Active Office. *IEEE Personal Communications*, 4, 42-47. 1997

[17] www.bluetooth.org: Specification of the Bluetooth System.

[18] UbiSense: http://www.ubisense.net

[19] Google Map: http://maps.google.com