Sap: Self Aware Protocol for Ubiquitous Object Communication

ThyagarajuG.S., Umakanth P. Kulkarni

Abstract*— The advances in computing technologies have resulted in an explosive growth in computing systems and applications [ubicomp] that impact all aspects of our life. People have an increasing desire for such ubiquitous access to information, anywhere, anyplace and anytime. This trend demands for unique protocol to establish a meaningful and useful communication across ubicomp objects. Self awareness property of the object driven automatically without human intervention makes their coexistence meaningful and is a necessary requirement of ubicomp systems. In this paper we are presenting a design and implementation of query language based self-aware protocol for ubicomp objects communication.

Index Terms*— self aware, protocol, ubicompobject, query language, bnf , sap.

I. INTRODUCTION

Self Awareness empowers individuals, as it gives them the ability to understand their position in a given system and use that knowledge to operate more effectively. Objects want to be aware of their own presentation, of what is appropriate in the given context, and how others perceive them. Objects produce immense quantities of data about their identity and behavior without an awareness of what that data is. Self awareness is a fundamental concern of the ubiquitous computing community, as awareness is necessary for interaction among the objects present in an environment. Self awareness for an object is all about, providing the complete information like: what the object is, in which environment the object is present ,which all other objects are present in that environment and what the object suppose to do in that environment. Some of the important aspects of self awareness are:

- To make the interactions among the objects possible.
- To make the objects more intelligent.
- To make the computing environment ubiquitous.
- To make the objects have complete information about themselves and about the environment in which they are present.

Language is mode of communication between two people. In the presented work we make the various objects to interact among themselves and with others. For this purpose, we make use of query language, which acts as a mode of communication between these objects. Any interactions among them or between the objects are possible through queries. The proposed query language handles data in the form of tree structure.

Manuscript Received October 15, 2011.

Thyagaraju.GS, Dept of CSE, VTU/SDMCET/ Dhavalagiri, Dharwad, Karnataka, India, Mobile Number: +919480123526, (e-mail: thyagaraju_gs@yahoo.co.in).

Umakanth P Kulkarni , Dept of CSE, VTU/SDMCET/ Dhavalgiri, Dharwad, Karnataka, India, Mobile Number : +919448915301 (e-mail: upkulkarnu@yahoo.com)..

Each node in tree represents an object (Ubicomp Object) having different attributes, like Manufacture ID, Object ID, Serial Number and Expiry Date.

The Query Language provides different queries that can be used to manipulate or retrieve data from the tree data base.

II. LITERATURE SURVEY

Different Technologies like RFID [1,5,6] and Electronic Product Code [EPC] used for automatic object identification are considered as the most emerging Ubiquitous Computing Technologies. Today RFID is a generic term for technologies that use radio waves to automatically identify people or objects. There are several methods of identification, the most common of which is to associate the RFID Tag unique identity with an object or person. The EPC system defines technical protocols and creates a data structure for stored information.

In a ubiquitous computing environment, all the objects present in the universe have to be given unique identity. The existing EPC method has some drawbacks [2]. These drawbacks are listed as follows-

- The number of bits allocated for each field in the tag is limited. As the number of objects grows, we need more and more bits to assign unique identifiers to each of these objects.
- There is no scope to add intelligence to the objects.

III. THE PROPOSED SELF AWARE PROTOCOL (SAP)

The proposed SAP (modified and integrated version of our earlier work [2,4]) is realized by addressing the following issues:

1. Object Self Awareness.
2. Object Categorization and unique object identity.
3. Query Language for object communication.

To increase the level of human comfort and security it is indispensable to categorize the objects as ubiquitous objects and assign each and every ubiquitous object a unique Id.

Ubiquitous object can be defined as an intelligent, autonomous computing object which can be accessed by anything at anytime and anywhere. Every ubiquitous object is identified by its unique object id (Ex: RFID tag). Each such ubiquitous object will be responsible for managing its own internal state, behavior and managing its interaction with other ubicomp compliant objects. One object after identifying other ubicomp compliant object near by, stores its identity in knowledge base. Depending on the type of the object and other relevant information the RFID tag indicates, the necessary actions will be initiated by the execution unit [2,4].

Every object can, in some sense, become smart by having RFID labels attached to them i.e. each object could acquire an electronic identity in addition to its physical structure. For example, an intelligent refrigerator may make use of the labels attached to the bottles, which could be useful in hotel rooms .The proposed ID format for RFID label content is shown in Fig1.Table1 gives the details of each field of the format.

A useful object classification scheme is necessary so that, the individual objects represent unique instances of larger classes and generic classes. The classification should also allow an active object to focus on the level of specificity that best suits its purpose. The proposed classification schema has the following features like scalability, consistent, complete, and responsive and supports drill down and roll-up features.

mfgId	#	objId	#	serNo	#	expDate

Figure 1: Format of RFID Label content.

Table 1: Details of the various Fields in RFID Tag

RFID Field	Description
mfgId	Variable length- Unique ID for Manufacturer. This is to be given by Designated Competent Authority.
objId	Variable length- Unique ID for objects Type. Ex- Food, Data Storage Device, Computational Device etc….This is to be given by Designated Competent Authority.
serNo	Unique serial number of that particular Object.
expDate	Object's expiry date.

The proposed object type consists of five level hierarchies for object classification and is:

1. **Logical Aggregate:** The logical aggregation of generic classes for analytical purpose.
2. **Category:** A collection of generic classes.
3. **Generic Class:** A commonly recognized group of interrelated classes.
4. **Class:** A group of objects sharing common use.
5. **Object:** A group of common functionalities.

XXX LOGICAL AGGREGATION
XXX CATEGORY
XXX GENERIC CLASS
XXX CLASS
XXX OBJECT

Figure 2: Hierarchical Object Classification

The scheme mentioned makes the active objects more intelligent. It provides the active objects with a capability to reason, analyze apparent relationships between objects in the hierarchy and hence facilitates the active objects in the decision making process.

Consider a situation where a user of an ubicomp compatible mobile enquires the ubicomp compatible refrigerator about the availability of a cold drink. If the cold drink is not available the active object will be able to inform the user that the cold drink is not available and will be able to provide an acceptable alternative, like available juice or milk. An example, illustrating the classification of objects is shown n fig3.

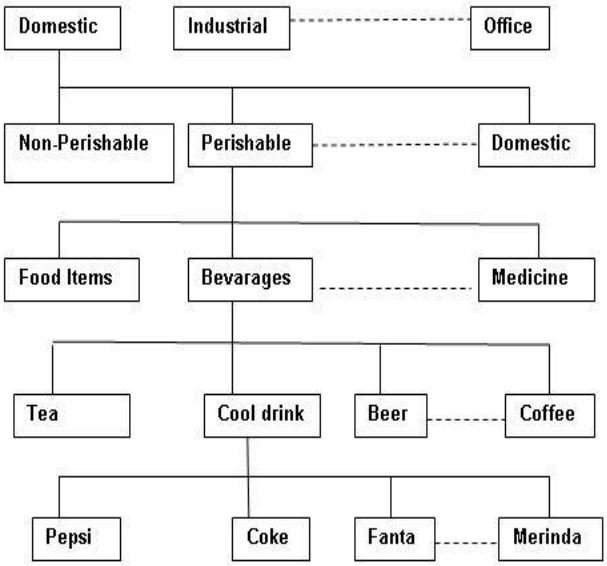

Figure3:Illustration of classification of Objects .

The 'expDate' is the programmable information, which determines the expiry date of ubicomp compliant objects under particular category. For example, a soft drink under object category of "liquid food item with Expiry Date" may contain the tag <Exp>13-06-2013</Exp> indicating the expiry date. Similarly many other standard tags may be defined as indicated in the table2 which facilitates the inter Ubicomp compliant object communication.

As an illustration assume that we have three ubicomp compliant objects namely PenDrive, Soft drink and a Medicine with RFIDs as:

1)PenDrive#1.1.3.4#aabbcc#<expDate>12-07-2011</expdate>, and

2)Pepsi#1.0.2.3#mmnnoo#<expDate>21-03-2014</expdate>,and

3)Crocin#1.0.3.2#eeffgg#<expDate>16-07-2015</expdate>.

These items are placed in an ubicomp compliant *refrigerator*, a smart intelligent element with the power of processing, here after termed as "active" ubicomp element. On the other hand pen-drive, soft drink and medicine are "passive" without any processing power. They only transmit their details to other objects. The refrigerator after reading RFIDs and identifying the object types of all three elements, may send message to the owner, informing that it is not right to place pen-drive in it or may inform the other details like expiry dates of the items or it may detect which of those items should not be taken together, in order to avoid harmful interactions or even irrelevant.

Table 2: Few other tags that can be included.

Tags/Commands	Meaning
<warDate> date </warDate>	Indicates warranty date
<SnI>	Switch on the device Immediately
<SoI>	Switch Off the Tag Immediately
<Ds> signature </Ds>	Indicates the digital signature for security purpose
<SoA> date/time </SoA>	Switch off the Devices at specified date and time

Query Language acts as a mode of communication between the objects. Any interaction among or between the objects is possible through queries. The Query Language provides different queries that can be used to manipulate or retrieve data from the tree data base. The query is parsed using the **lexer** (lex program) and **parser** (yacc program)[3,4].

Here the grammar for yacc program is implementation of the **BNF** grammar given below:

QL_statement::= getInfo_statement | add_statement | delete_statement
getInfo_statement::= getparent_clause | getdetails_clause |
getsiblingcount_clause
| getsiblingdetails_clause
| getchildren_clause | getlevel_clause |
getdepth_clause | getleafnodes_clause;
add_clause ::= ADD '(' STRING ',' STRING ',' STRING ','
DATE ')' ';'
delete_clause::= DELETE del_where_clause ';'
getparent_clause:= GETP WHERE NAME EQ STRING ';' ;
getdetails_clause::=GETD WHERE predicate ';' ;
getsiblingcount_clause::= GETSC WHERE NAME EQ STRING ';'
;
getsiblingdetails_clause ::= GETSD WHERE NAME EQ STRING ';'
;
getchildren_clause ::= GETC WHERE NAME EQ STRING ';'
;
getlevel_clause ::= GETL WHERE NAME EQ STRING ';' ;
getdepth_clause ::= GETDT WHERE NAME EQ STRING ';' ;
getleafnodes_clause ::= GETLN WHERE NAME EQ STRING ';'
where_clause ::=| WHERE search_condition ;
del_where_clause ::= WHERE search_condition ;
search_condition::= predicate ;
predicate ::= NAME EQ STRING | ED COMPARISION DATE | ED EQ DATE;
ADD::=ADD | add | Add
DELETE::= DELETE | delete
GETP::= GETPARENT | Getparent | getParent | getparent
GETD::= GETDETAILS | Getdetails | getDetails | getdetails
GETSC::= GETSIBLINGCOUNT | Getsiblingcount | getSiblingcount | getsiblingcount
GETSD::= GETSIBLINGDETAILS | Getsiblingdetails | getSiblingdetails | getsiblingdetails
GETC::= GETCHILDREN | Getchildren | getChildren | getchildren
GETL::= Getlevel | getLevel | getlevel
GETDT::= GETDEPTH | Getdepth | getDepth | getdepth
GETLN::= GETLEAFNODES | Getleafnodes | getLeafnodes | getleafnodes
WHERE::= WHERE | where

Table below illustrates some examples how to write the queries in order to establish the communication between ubiquitous objects.

Table3: Syntax and examples of few queries

Query	Syntax / Example
Add	Add(mfgId,objId,serNo,Date); / Add("apple","0.0.0.0","123",12-11-2011);
Delete	Delete where condition; / Delete where mfgId="apple";
GetDetails	GetDetails where condition; / GetDetails where mfgId="apple";
GetsiblingCount	GetsiblingCount where condition; / GetsiblingCount where mfgId="mango";
GetsiblingDetails	Getsiblingdetails where condition; / GetsiblingDetails where objId="0.0.0.1.0";
GetChildren	GetChildren where condition; / GetChildren where objId="0.0.0.1.0";
GetLevel	GetLevel where condition; / GetLevel where serNo="123";
GetDepth	GetDepth where condition; / GetDepth where serNo="123";
GetLeafNodes	GetLeafNodes where condition; / GetLeafNodes where objId="0.0";

IV. MODELING AND DESIGN OF PROPOSED ARCHITECTURE

Various system components and their coexistence along with their behavior is shown in Fig4.

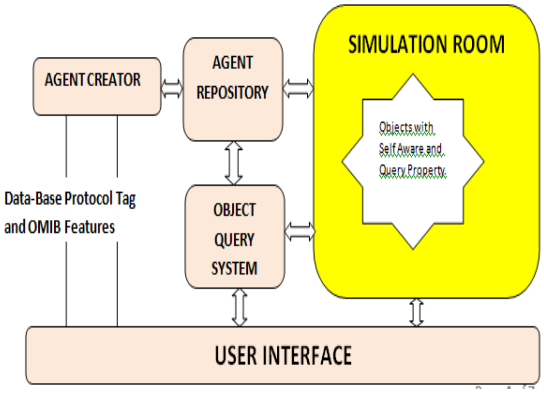

Figure 4: Architectural Design

Agent Creator: Allows creation of various objects and assigns the property

Protocol & OMIB (Object Management Information Base) Unit : Standard hierarchical storage and operation on storage, Tag Repository and usage in the specific protocol version.

Simulation Room : Meeting place for all Objects and exercise the protocol capabilities under simulated environment

Coordination Environment- GUI : Main program which encapsulate underlying all system capabilities.

Object Querying System: Subsystem to simulate querying and information retrieval features.

Fig5 illustrates the working of SAP. The protocol is validated using three agents' refrigerator, owner and passive objects.Fig6 and fig7 shows the use cases of owner and refrigerator objects.Fig8 illustrates the different classes and their association used for implementing SAP.

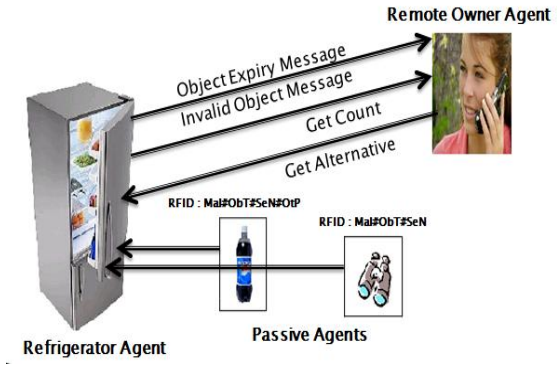

Figure5: Brief Working of Protocol SAP

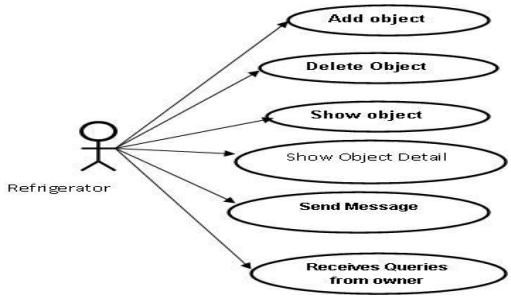

Figure6: Use-Case diagram for the Refrigerator Object

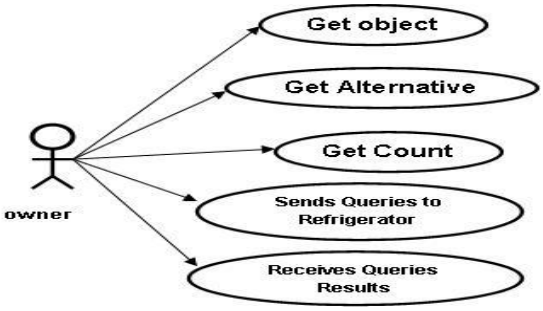

Figure 7: Use-Case diagram for the Owner Object

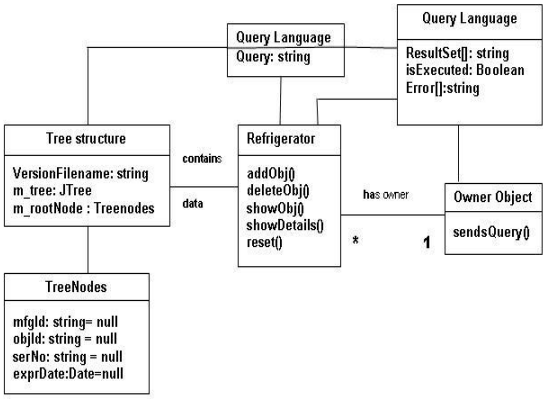

Figure8: Class Diagram of Self Aware Protocol

V. IMPLEMENTATION AND EXPERIMENTAL STUDY

The simulator for the proposed system is implemented using Java Software Development Kit, Eclipse and Linux Platform (Fedora version 14). The proposed SAP is validated using refrigerator, owner object and passive objects.

Refrigerator: The refrigerator object is a active object which can communicate to owner all the details about the objects present in it making use of its knowledge base. The refrigerator object periodically accepts the RFID signals emitted by the objects and processes these RFID signals and depending upon the result of the processed information it sends various messages to its owner object. The refrigerator object can also receive messages from the owner object. The simulated refrigerator GUI is shown in fig9, which includes dialog boxes to create objects, delete objects, view object details and to view the various activities that are going inside it like messages sent to owner object and the messages received from the owner object.

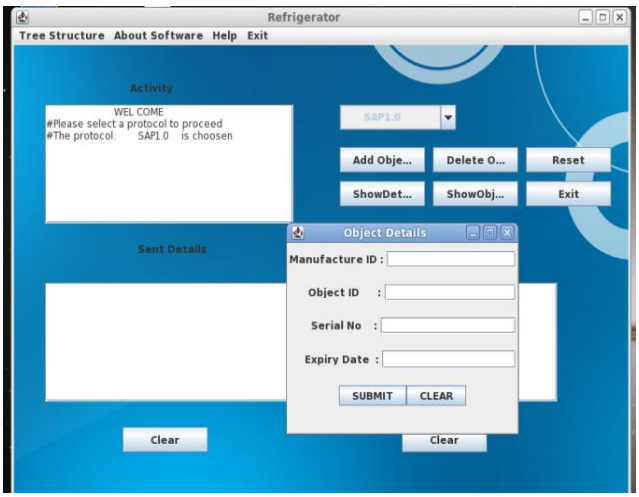

Fig9:G.U.I for Refrigerator Object

Owner Object: The owner object is also a active object which can communicate with refrigerator object. Fig10 illustrates the GUI of owner object which includes the dialog boxes to know the objects present in the refrigerator, to submit a query and to display the received messages .It can send messages to the refrigerator like get Count-to get the number of objects of a specified type, get alternative-to get an alternative for a specified object, get Objects- to get the objects hierarchy in the application.

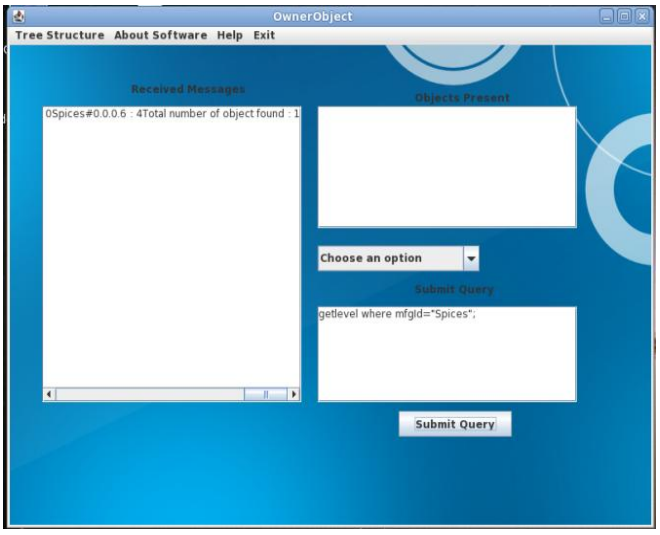

Figure 10: G.U.I for Owner Object

Passive Object: This class only emits RFID signals. The RFID signals contain various self information about the object like Manufacture ID, Object ID, and serial no. and expiry date. As this object does not do any other work, no GUI is provided for this object.

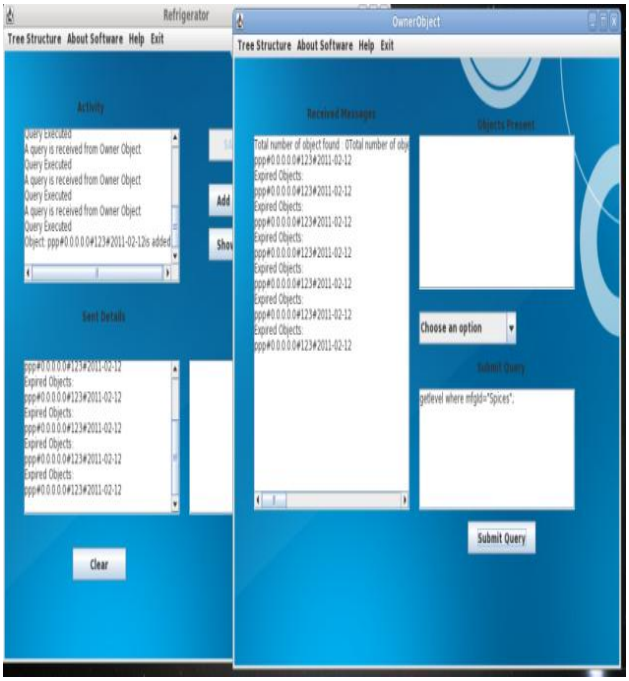

Fig11: Illustration of communication between owner object and refrigerator.

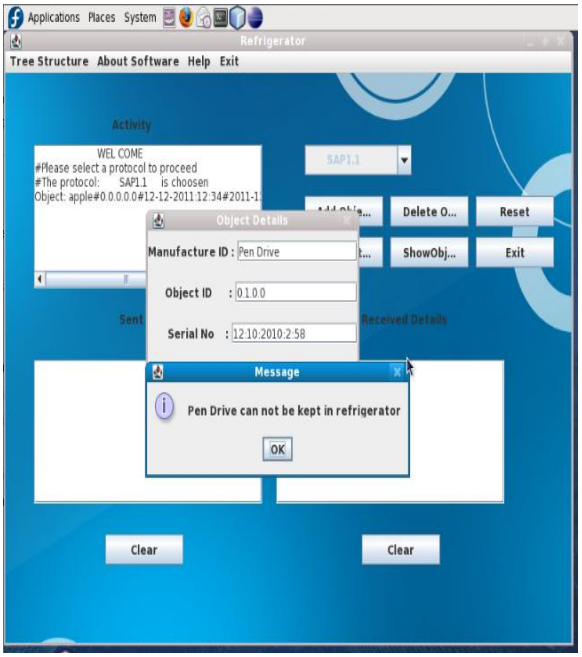

Fig12: Illustration of object incompatibility

Experiment was conducted by considering different category of objects like domestic and industrial. In domestic category we considered perishable (food items ,fruits, vegetables , grains, milk products , green leaves ,bakery items ,beverages, medicines, cosmetics and cleansing agents), non perishable (electronic appliances, plastics ,toys and ornaments). In industrial category we considered objects like chemical and daily usable objects. In Total about 50 passive objects was considered .Using SAP communication was established among these objects. Table 4 gives the list of messages that was communicated successfully among the objects.

Table4: Messages communicated successfully

Sl.No	Messages communicated successfully
1	Refrigerator is Empty
2	Refrigerator is Full
3	Object is Incompatible
4	Object is Expired
5.	Object is about to Expire
6	Object is in Good Condition
7	Object Details
8	Alternative Object is present
9	List of objects
10	Object Count
11	Door is open.
12	Incompatible Object is present
13	Object is Compatible
14	Refrigerator details
15	List of objects expired
16	Object is present
17	Objects expiry date
18	Delete Object
19	Add object

VI. CONCLUSION

The protocol designed facilitates a communication of very basic details of the object (like name, id , expiry date and compatibility) . The SAP can be further improved by adding more programming related attributes like temperature at which objects can be kept, calorie contents of the object, warranty date of the object, user registration, human health factor, etc. More query commands can be added to the Query Language to enhance the efficiency of interaction among the objects.

VII. ACKNOWLEDGEMENT

The authors are thankful to the following final year students(2011), Premalata G Langati, Priyanka K and Ruheen Banu Gadwale of final year engineering, for their contribution in providing softwares and simulator towards the implementation and experimental verification of the proposed SAP.

REFERENCES

1. C.M. Robert, "Radio Frequency Identification (RFID), Elsevier Journal: Computer & Security, pp18-26-2006
2. U.P.Kulkarni et al " Ubiquitous Object Categorization and Identity", International Conference on Computational Intelligence for Modeling Control and Automation, and International conference on Intelligent Agents, Web Technologies and Internet Commerce(CIMCA-IAWTIC' 06)- IEEE2006
3. lex & yacc by John Levine,Tony Mason and Doug Brown
4. U.P.Kulkarni,etal "Query Language Interface for Ubiquitous Objects" , International Conference on Computational Intelligence and Multimedia Applications 2007,IEEE , DOI 10.1109/ICCIMA.2007.331
5. M Satyanarayanan "Pervasive Computing: Vision and Challenges".,School of Computer Science Carnegie Mellon University – IEEE Personal Communication, 2001.
6. Anthony D. Joseph, Almudena Díaz, Pedro Merino, F. Javier Rivas, Umakant P. Kulkarni, J.V.Vadavi, G.S. Thyagaraju, S.M. Joshi, and A.R. Yardi, "Mobile and Ubiquitous Objects" ,IEEE pervasive computing journal ,vol5 ,No3, July–September 2006 ,wips ,pg 57-60.

AUTHORS PROFILE

Thyagaraju.GS received the M.Tech Degree in Computer Science And Technology From University Of Mysore ,India in 2002.He has got ten years of experience in academics ,fours years of Research Experience . He is a member of IETE.He has guided many students at UG and PG level.He is pursuing Ph.D in Computer Science Engineering. His Research Interests are Context Aware Computing in Ubiquitous and Intelligent Systems He is now working as a Senior Lecturer in Dept Of CSE, SDM College Of Engineering, Karnataka, Dharwad.He can be reached at thyagaraju_gs@yahoo.co.in.

Dr. Umakant Kulkarni obtained his BE Degree from Karnataka University, Dharwad in the year 1989, ME Degree from PSG College of Technology, Coimbatore in the year 1991 and PhD from Shivaji University, Kolhapur in the year 2007. He has published many papers at International Journal and IEEE conferences in the areas of Pervasive and Ubiquitous Computing, Distributed Data Mining, Agents Technology and Autonomic Computing. He is Member of IETE and ISTE. He served as Head of Department and Chief Nodal Officer- TEQIP a World Bank funded project. He has guided many students at PG level and five research scholars are pursuing their PhDs. Currently he is serving as professor in the Department of Computer Science & Engineering, SDM College of Engineering & Technology, Dharwad, Karnataka State, India. He can be reached at upkulkarni@yahoo.com