SmartSense: A Novel Smart and Intelligent Context-Aware Framework

Anuja Meetoo-Appavoo
Faculty of Engineering University of Mauritius

Abstract: Traditional interactive applications are limited to using only explicit user input. However, as users are moving away from traditional desktop computing environments towards mobile and ubiquitous computing environments, there is an increasing need for such pervasive applications, which operate in an extremely dynamic and heterogeneous environment, to be context-aware. Yet, context is poorly utilized and building context-aware applications is currently complex and time-consuming mainly due to a lack of architectural support. There is currently no established generic architecture and most context-aware applications have been built in an ad hoc manner. This paper presents SmartSense, a smart and intelligent context-aware framework for the easy creation and flexible deployment of context-aware applications, and hence helps in transforming physical spaces into computationally active and intelligent environments. In addition to fully supporting all the requirements of a context-aware architecture, it (1) allows applications to easily acquire contextual information, reason about it using different logics and then adapt themselves to changing contexts, (2) allows autonomous heterogeneous components to have a common semantic understanding of contextual information and aids in developing more scalable and interoperable applications through the use of ontologies, (3) provides an improved Quality of Context (QoC) for location and seamless indoor and outdoor location tracking through sensing fusion, (4) employs machine learning, and (5) is generic, i.e. will support any context-aware application, and scalable.

Keywords: Context-awareness, framework, architecture, middleware, sensing fusion, machine learning.

INTRODUCTION

With the infiltration of mobile devices and mobile communication to support a mobile lifestyle, pervasive computing is becoming increasingly popular. Computing is moving away from desktop and permeating into several everyday objects and the environment in which we live thus allowing services to be provided to users anywhere and at anytime. Traditional interactive applications are limited to using only explicit user input. However, the new generation of applications operates in an extremely dynamic and heterogeneous environment where the availability of resources and services may change significantly during a typical period of system operation. Thus, applications must be context-aware and dynamically adapt to changes in their environment due to activities of the users or other objects. Today, context-awareness is receiving growing attention and has moved beyond its research roots.

Yet, despite context being vital and the prevalence of powerful networked computers that makes feasible the use of an increasing number of commercial off-the-shelf sensing technologies, context is a poorly utilized source of information in interactive computing. Building context-aware applications is currently complex and time-consuming mainly due to a lack of architectural support. There is currently no established generic architecture and most context-aware applications have been built in an ad hoc manner. In this paper, SmartSense, a novel smart and intelligent context-aware framework that supports the development and evolution of context-aware applications is proposed. The main aim of SmartSense is to make physical spaces, such as rooms, homes, offices and shopping centers, intelligent and eventually aid humans in these spaces.

Section 2 gives a technical discussion on context and context-awareness. Section 3 presents the requirements of a context-aware framework and evaluates some related works with respect to the requirements. Section 4 presents SmartSense, my proposed context-aware framework. Section 5 gives an overview of a proof-of-concept application built on SmartSense, namely a child tracking service. Section 6 provides an evaluation of SmartSense, outlining its strengths and benefits. Finally, section 7 concludes the paper, outlining some future works.

CONTEXT AND CONTEXT-AWARENESS

Apprehending the need for context [1] is only the initial step towards using it effectively. However, a better understanding of context will enable application designers to more effectively select what context to use in their applications and provide insights into the types of data that need to be supported and the abstractions and mechanisms required to support context-aware computing. The method of context acquisition [2] is of paramount importance when designing context-aware systems since it predefines the architectural style of the system at least to some extent. Some examples of on-going research projects in the field of context-awareness are discussed in [3] - [11] and common context-aware applications are outlined in [12] - [15]. A context model [2] defines and stores context data in a form that can be processed by

Manuscript received August 5, 2011
Manuscript revised August 20, 2011

machines. Early models [16] primarily dealt with modeling of context with respect to one application or an application class. However, generic models that cater for several applications are desired. The most relevant context modeling approaches [2] are (1) key-value model [2], (2) markup scheme model [2], (3) graphical model [2], (4) object-oriented model [2], (5) logical model [2], and (6) ontology based model [17]. These approaches are based on data structures used for representing and exchanging contextual information in the respective system. Moreover, the ontology based model is the most expressive and fulfils largely the requirements [2] defined in [16].

Quality of Context (QoC) [18] refers to information that describes context information and can be used to determine the worth of the information. Sensing a lot of redundant and conflicting information [19] makes efficient management of context information challenging. QoC parameters can then be used to resolve conflicts in context information. However, only a few works [20] support quality of context in context-aware applications.

CONTEXT-AWARE FRAMEWORK

Context is vital for the new generation of applications where the user's context, e.g. the user's location and the people and objects around him/her, is dynamic. Moreover, recent advances in technology, namely in sensors, hardware, networking and software, have made feasible the development of context-aware applications.

Yet, context is poorly utilized due to a few properties of context [1], namely (1) context is acquired from non-traditional devices with which humans have limited experience, (2) context must be abstracted to make sense to applications, (3) context may be acquired from multiple distributed and heterogeneous sources, and (4) context is dynamic.

However, despite these problems, researchers have been able to develop context-aware applications. But an ad hoc process has been employed making the development of context-aware applications complex and time-consuming. This lack of architectural support makes it hard to build new applications and limits the amount of reuse across applications, requiring common functionalities to be rebuilt for every application. Therefore, the main problem preventing the use of context and development of more context-aware applications is a lack of architectural support. Existing architectures [21] fail to provide the necessary support for adaptive context-aware applications leading to the need for context-aware framework.

The following are the most crucial prerequisites of a context-aware framework:

i. *Context specification* [1], [22]: A context-aware framework must provide the flexibility to context-aware applications to query it or subscribe to it for required context information.
ii. *Separation of concern and context handling* [1]: Separating context acquisition from context use will help reduce the design process of acquisition and enable applications to use context information without the need to worry about sensor details and how to acquire context from it.
iii. *Context interpretation* [1], [22]: Context interpretation must be provided by the framework to make it reusable by multiple applications. It must be independent of applications using it.
iv. *Constant availability of context acquisition* [1], [22]: A component responsible for acquiring context must be independent of applications using it such that context is constantly available to any application requiring it. This will also relieve application designers from the concerns of context acquisition components.
v. *Transparent distributed communications* [1]: Context-aware applications and sensors may be distributed among several computing devices. However, this must be transparent to both sensors and applications, thus freeing designers from the need to build a communications framework.
vi. *Context storage* [1], [22]: Past context information must be made available to context-aware applications as well as components of the architecture, e.g. to predict future context values.
vii. *Resource discovery* [1], [2], [22]: Context sources are not stable or permanently available. In such a dynamic environment, resource discovery is vital. This feature also enables designers to discover what context and components are already available in the environment at design time.
viii. *Model of the environment* [22]: Availability of such a model to the framework and applications allows for more sophisticated inferencing.
ix. *Machine learning*: Rule based inferences have the disadvantage of requiring explicit rule definitions by humans and do not provide the flexibility of adapting to changing circumstances. Machine learning techniques help to deduce higher level context and hence cater for this shortcoming.

There are currently a diversity of near-context-aware architectures [1], [22], context-aware architectures [1], [22] – [25] and context-aware middleware [26] – [35]. However, they all have loopholes and each is restricted to a specific domain. Furthermore, none of them fully supports all the requirements of a context-aware framework that have been identified, as shown in table 2 [36]. Thus, there is no generic established context-aware architecture to aid in building context-aware applications.

Table I. Summary of Existing Architectures and Middleware

Requirements / ***Existing Framework***	***Context Specification***	***Separation of Concern & Context Handling***	***Context Interpretation***	***Constant Availability of Context Acquisition***	***Transparent Distributed Communications***	***Context Storage***	***Resource Discovery***	***Model of the Environment***	***Machine Learning***
Near-Context-Aware Architectures									
Open Agent Architecture	P	✓	✓	X	✓	X	P	X	N/A
HIVE	P	P	X	✓	✓	X	✓	X	N/A
MetaGlue	P	✓	X	✓	✓	X	✓	X	N/A
Context-Aware Architectures									
Context Toolkit	✓	P	✓	✓	✓	✓	✓	X	X
Sulawesi	P	P	X	✓	X	X	P	✓	N/A
Stick-e Notes	P	✓	X	X	X	X	P	X	N/A
EasyLiving	P	✓	P	✓	✓	X	P	P	N/A
Schilit's System Architecture	P	P	X	✓	P	X	P	✓	N/A
CALAIS	✓	P	X	✓	✓	X	✓	N/A	N/A
TEA	P	✓	✓	X	X	X	P	N/A	N/A
SAIsense	✓	✓	✓	✓	✓	✓	✓	✓	X
Context-Aware Middleware									
CASS	✓	✓	✓	N/A	✓	✓	X	X	N/A
SOCAM	✓	N/A	✓	N/A	✓	✓	✓	X	N/A
MiddleWhere	✓	✓	✓	N/A	N/A	✓	X	✓	N/A
Gaia	✓	N/A	✓	N/A	N/A	✓	✓	X	✓
QoSDREAM	✓	✓	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CAPNET	N/A	N/A	N/A	N/A	✓	✓	✓	X	N/A
CoBrA	✓	✓	✓	N/A	N/A	✓	N/A	X	N/A
Contory	✓	✓	✓	✓	✓	✓	✓	X	N/A

X = No Support; P = Partial Support; ✓ = Complete Support; N/A = Not Available

SMARTSENSE: A NOVEL SMART AND INTELLIGENT CONTEXT-AWARE FRAMEWORK

SmartSense eases the development of new context-aware applications as well as the evolution of existing ones, e.g. to seamlessly change the current sensing techniques used. At the core of the framework is an inference engine, based on the predictive model of context defined in the ontology, to make inferences about the current context and help context-aware applications in determining how to adapt their behavior when new context information is acquired. A key feature of the framework is that it employs machine learning to deduce higher level context and provides the flexibility of adapting to changing circumstances, hence making SmartSense smart and intelligent. Furthermore, SmartSense is server-based and hence scalable, i.e. remains effective with an increase in the number of users and services and new components can be added as and when needed. It also implies that SmartSense does not suffer from processor and memory constraints that would apply to a mobile computer. This makes feasible the use of artificial intelligence (AI) components and the storage of large amounts of data, namely context history, facts and inference rules. The architecture employs ontologies to model context, namely the Web Ontology Language (OWL) to encode facts and the Semantic Web Rule Language (SWRL) to encode rules, thus making context-aware applications interoperable and scalable. The sensing fusion algorithm improves the QoC for location and makes the framework *adaptive* by granting the flexibility to alter location sensing techniques on the fly without affecting applications or infrastructure components dependent on location context. As such, the architecture supports seamless indoor and outdoor location tracking.

Fig. 1 depicts the overall architecture of SmartSense and is followed by a brief description of each of its components. The components are implemented in Java and are multithreaded. Thus, SmartSense is also platform-independent and its components are capable of handling multiple incoming messages.

A. Context Widget

Context widget, that is similar to the context widget in Context Toolkit [23], acts as a mediator between an application and its operating environment. It relieves applications from context sensing issues by wrapping the hardware or software sensor with a uniform interface. This provides the flexibility of exchanging a context widget with another widget providing the same type of context information without affecting the applications using it. Furthermore, context widgets (1) can be shared by all executing applications and are independent of these applications, (2) are reusable and persistent, and (3) continuously send updated context information to the context manager for context encoding and storage and further processing when needed. Thus, context widget caters for (i) context specification, (ii) separation of concern and context handling and (iii) constant availability of context acquisition. Context information is made available through querying or notification mechanisms via a common, uniform interface.

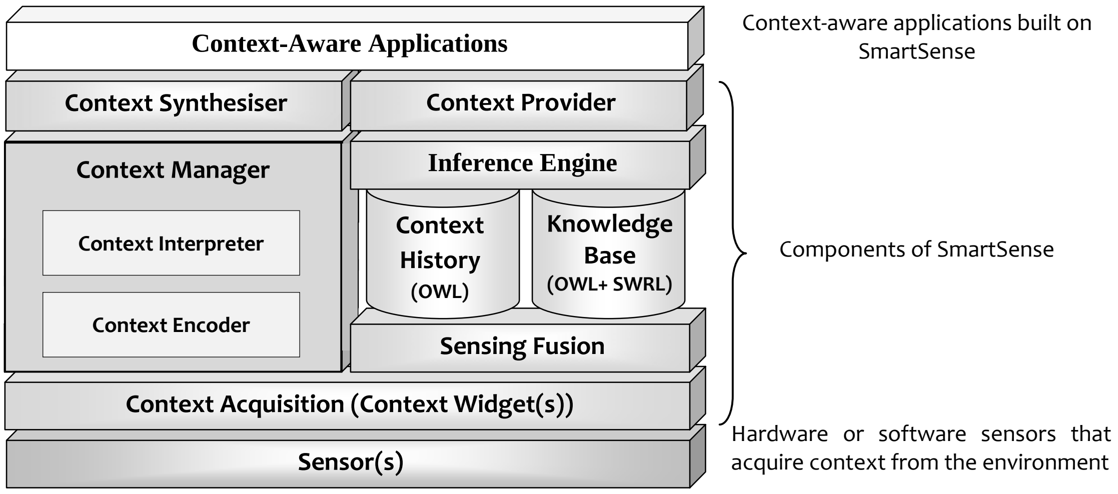

Context-Aware Applications
Context-aware applications built on SmartSense
Context Synthesiser
Context Provider
Context Manager
Inference Engine
Context Interpreter
Context History (OWL)
Knowledge Base (OWL+ SWRL)
Context Encoder
Sensing Fusion
Components of SmartSense
Context Acquisition (Context Widget(s))
Sensor(s)
Hardware or software sensors that acquire context from the environment

Figure 1. SmartSense framework

B. *Context Manager*

The context manager comprises of two subcomponents, namely:

(1) Context interpreter: The context interpreter derives higher level or richer forms of information from low-level or raw context information acquired from context widgets. It is independent of applications and can be used by multiple applications. The latter need not be concerned about the procedures involved but only interact with the interpreted information. The context interpreter works by invoking the inference engine.

(2) Context Encoder: The context encoder encodes context information acquired from sensors through context widgets using OWL and stores it in context history. In cases where a running application is dependent on the context, the context manager sends the URI of the saved context to the context provider. Thus, the context manager caters for (i) context interpretation.

C. *Sensing fusion*

A plethora of location sensing technologies is available and no single technique has emerged as the clear winner in all types of environment. Instead, the use of different location sensing technologies [37] is expected in different environments depending on their specific requirements. Some environments may also require the deployment of multiple location technologies. However, different location sensing techniques give location information in different formats and with different resolution and confidence. SmartSense employs a sensing fusion algorithm [38] to (i) fuse location context acquired from any number of different location sensing techniques using probabilistic reasoning techniques to obtain an optimal estimate of the context information, thus improving accuracy and positioning probability, (ii) resolve conflicts, and (iii) handle uncertainty. It also enables seamless transition from one location sensing technique to another.

D. *Context history*

Applications may be dependent on current context as well as past context to adapt their behavior. Hence, context acquired from sensors are encoded in Web Ontology Language (OWL) [39] – [41], more precisely OWL DL, and stored in context history even when no applications currently require them. Context history can be queried by applications and thus caters for (i) context storage.

E. *Knowledge Base*

Knowledge base contains facts encoded in OWL and rules encoded in Semantic Web Rule Language (SWRL) [41] – [43]. It is used by the inference engine to infer new facts and solve problems of context-aware pervasive applications. The knowledge base is tailored to each application built on SmartSense and can be altered or updated without affecting the implementation of the inference engine.

F. Inference Engine

The inference engine Bossam version 0.9b5 [42], developed by M. Jang, is used to find matching goals when a change in context is detected. Applications can then adapt their behaviors in-line with the inference. It employs inference rules to perform context reasoning over stored facts and works in conjunction with a knowledge base that stores facts and rules, and a context history that stores past and current context as facts. The framework also caters for reasoning and learning mechanisms that allows for more accurate prediction of context and hence adaptation of the application. Thus, it caters for (i) machine learning.

G. Context Provider

The context provider [36] accepts subscriptions from context consumers, keeps record of subscribers for particular context types and issues a callback to them when updated context is available from the context manager. While context widgets provide applications with raw context information, context providers employs reasoning and learning mechanisms [44] to reason about context and offers context encoded in OWL. Thus, it caters for (i) context specification. Different context providers may use different reasoning or learning mechanisms, hence logics. Yet, their common grounding on the predicate model allows for a uniform query by context consumers. All context providers support a similar interface allowing consumers to issue queries for context information without worrying about the type of the context provider, hence greatly simplifying the development of context-aware applications.

H. Context Synthesiser

A context synthesizer acquires sensed context from various context providers, deduce higher-level or abstract context from them, and provide the deduced context to consumers. For example, a context synthesizer can infer a child’s activity based on the number of children in the room and the applications that are running. Context synthesizers accept subscriptions from context consumers, keep records of subscribers for particular context types and issue a callback to them when updated context is available from the context manager. Reasoning and learning mechanisms, based on Bayes theorem [44], are also employed. All context information sensed as well as user actions are stored in the context history. These act as training examples for the framework. This approach is especially useful for learning users’ behaviours by studying their actions over a period of time. Eventually, applications can even take proactive actions on behalf of the user depending on the context and thus save valuable time of a user. Thus, it caters for (i) context specification and (ii) machine learning.

I. Communication between the Components of the Framework

Components of SmartSense are autonomous in execution, i.e. they are instantiated and execute independently of each other. Context-aware applications are usually distributed on several computers for better performance and efficiency. SmartSense allows a peer-to-peer communication among them using HyperText Transfer Protocol (HTTP) and eXtensible Markup Language (XML). Messages are encoded in XML and wrapped with HTTP. These allow for the lightweight integration [1] of distributed components, enable the architecture to benefit from the platform- and language-independence features of XML and enable to build more interoperable context-aware services. Two classes, namely XMLHTTPClient and XMLHTTPserver, are provided to send and receive messages respectively. Application programmers need not write these classes, but merely create instances of them to use them. The default communication protocol for SmartSence can be modified to account for other protocols, e.g. Simple Mail Transfer Protocol (SMTP), simply by creating an object that speaks the SMTP protocol for outgoing communications and one for incoming communications. Thus, it supports (i) transparent distributed communications.

J. Resource Discovery

Resource discovery mechanisms [5] are rarely used in existing frameworks. However, such dynamic mechanisms are important, particularly in pervasive environment, where available sensors and the context sources change rapidly. In real world applications, used context sources are neither stable nor permanently available. SmartSense exploits Jini [45] - [47] that allows service registration by context providers and service discovery by context consumers. All components of SmartSense and applications built on it register themselves to the discovery service, which provides information required to communicate with them at run-time. This caters for (i) resource discovery.

Prototype Application Built on the Proposed Framework

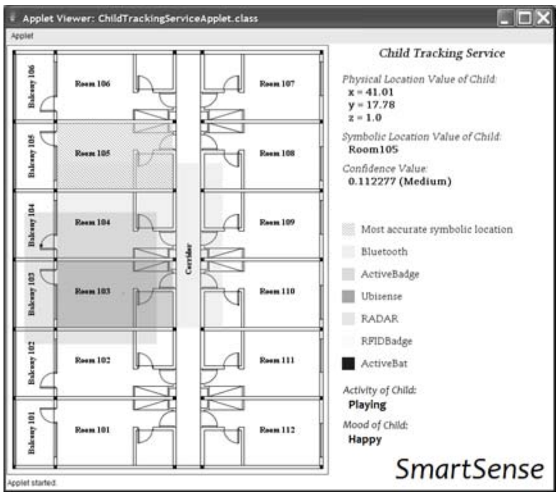
A child tracking service has been implemented on the framework. The service allows parents to track their children by viewing the latter’s location on their portable device as well as getting a prediction of the child’s activity, e.g. whether he is playing around or watching a film, and mood, e.g. whether he is happy or sad. It uses location context and status of applications in the room, exhibits the use of the components of SmartSense and illustrates the sensing fusion algorithm and learning mechanisms of the framework. The application currently deals with part of a building and can be extended to any building by altering the stored model of the environment.

SmartSense components used in this application are *widgets providing location context and the status of applications*, *sensing fusion algorithm*, *physical interpreter*, *symbolic interpreter*, *physicalToSymbolic interpreter*, *context encoder*, *context provider*, *inference engine*, *context history* and *knowledge base*. A screenshot of the application is given in fig. 2 where location values from six sensors are fused to obtain an optimal location value for the child being tracked.

Examples of rules for the ChildActivity context provider are as follows:

Child(sitting) AND Television(On)
=> ChildActivity(Watching Movie)
People(Room, >, 1) AND NOT Child(sitting)
=> ChildActivity(Playing)

Sensing of the child's mood [44] of is very challenging since it is difficult to write rules to determine the mood of a person as each person is different and a large number of factors can influence a person's mood. The child tracking application employs a learning mechanism, based on the Naïve Bayes algorithm, that takes into account factors like the location of the child, time of the day, which other persons are in the room with him and his activity to determine the child's mood. Past context is used to train the components of the framework for some time.



Physical Location Value of Child:
x = 41.01
y = 17.78
z = 1.0
Symbolic Location Value of Child:
Room105
Confidence Value:
0.112277 (Medium)
Activity of Child:
Playing
Mood of Child:
Happy

Figure 2. Screenshot of Child Tracking Service

EVALUATION

SmartSense is a hybrid solution and greatly facilitates the development of any context-aware applications. A child tracking service has been easily implemented using the framework. SmartSense allows applications to subscribe to context widgets or context providers for context types they require and receive notifications when updated context values are acquired. This eliminates the need for the applications to query the widgets at regular intervals for new context values. It also hides the details of context sensors, thus allowing the underlying sensors to be replaced without affecting the applications dependent on context they provide. Moreover, the context widgets are independent of applications and can thus be shared among several applications. It has features of both a context-aware architecture and a context-aware middleware. Moreover, SmartSense makes use of machine learning and thus allows the flexibility of adapting to changing circumstances. It has been found that prediction of the mood of the child was fair in different situations after some training, given that humans [44] are fairly repetitive creatures, i.e. their moods in different contexts follow certain predictable patterns. It should however be noted that predictions may not always be perfect since it is quite difficult to take into account all possible factors that can influence the mood of the user. Good guesses can be made based on the available information.

Furthermore, SmartSense supports all the requirements identified for a context-aware framework. The sensing fusion algorithm of SmartSense improves the QoC for location by refining the location context values acquired from any number of location sensors, each employing a different location sensing technique. The algorithm also makes it very simple to dynamically change the underlying location sensing technique at run time, and thus caters for seamless indoor and outdoor location tracking. SmartSense also supports the lightweight integration of components though the use of XML and HTTP for communication. XML provides platform- and language- independence. Moreover, the components are implemented in Java rendering the architecture platform independent. Interpreters provided by SmartSense are independent of and can be shared among applications. In addition, the use of ontologies to model context enables more scalable and interoperable applications to be built. Bossam, the inference engine used by SmartSense, makes use of URI to refer to files which makes it suitable for use with the distributed infrastructure components.

CONCLUSION

This paper has presented a set of requirements identified for a context-aware framework and a study of existing relevant near-context-aware architectures and on-going existing context-aware architectures and context-aware middleware. It has been shown that none of the existing architectures fully satisfies all the requirements. Thus, SmartSense, a novel smart and intelligent context-aware framework has been proposed and implemented. A proof-of-concept application, namely a child tracking service, has been implemented using the framework to illustrate that the latter facilitates the development of context-aware applications. While SmartSense supports all requirements of a context-aware framework, there are still rooms for improvement to make SmartSense a commercially acceptable framework. Some issues that merit further investigation are (i) improving the sensing fusion algorithm to take into account freshness of context values by associating a time-to-live value in seconds with each sensor value, after which the value is considered stale and unusable, (ii) allowing an infrastructure component that went down to automatically restart any unfinished jobs when the system works again, e.g. the context provider can periodically save the current list of subscribers to permanent storage so that it can be restored to its last working state when restarted, (iii) enabling the architecture to detect failure of a component and automatically restart it as well as restore it to its last working state, (iv) providing a graphical interface to aid the developer in constructing inference rules by presenting him with a list of the various types of contexts available and a list of possible behaviours of the application for the different context, (v) improving the learning mechanisms, such making use of reinforcement learning.

References

[1] A. k. Dey, Providing architectural support for building context-aware applications, PhD Thesis (Georgia Institute of Technology), 2000.

[2] M. Baldauf, S. Dustdar, & F. Rosenberg, A survey on context-aware systems, *International Journal of Ad Hoc and Ubiquitous Computing*, 2006.

[3] K. L Mills, AirJava: Networking for smart spaces, *Proc. USENIX Embedded Systems Workshop*, Cambridge, Massachusetts, 1999, 29 – 34.

[4] Information Society Technologies Research Group, The disappearing computer. [Online] Available from: http://www.disappearing-computer.net [Accessed 2011].

[5] Electrical Engineering and Computer Science Department, University of California, Berkeley, The endeavour expedition: Charting the fluid information utility. [Online] Available from http://endeavour.cs.berkeley.edu [Accessed 2011].

[6] Carnegie Mellon University, Project Aura, Distraction-free ubiquitous computing. [Online] Available from: http://www.cs.cmu.edu/~aura [Accessed 2011].

[7] MIT Laboratory for Computer Science and MIT Artificial Intelligence Laboratory, MIT Project Oxygen - Project overview. [Online] Available from: http://oxygen.lcs.mit.edu [Accessed 2011].

[8] Department of Computer Science and Engineering, University of Washington, Portolano: An Expedition into Invisible Computing. [Online] Available from: http://portolano.cs.washington.edu [Accessed 2011].

[9] Microsoft Research, Easy Living. [Online] Available from: http://research.microsoft.com/easyliving [Accessed 2011].

[10] S. Shafer, J. Krumm, B. Brumitt, B. Meyers, M. Czerwinski, & D. Robbins, The new EasyLiving project at Microsoft Research, *Joint DARPA/NIST Smart Spaces Workshop*, Maryland, 1998.

[11] M. Lucente, IBM Research, DreamSpace: natural interaction. [Online] Available from: http://www.research.ibm.com [Accessed 2011].

[12] G. Chen, & D. Kotz, A Survey of context-aware mobile computing research, *Technical Report TR2000-381 (Department of Computer Science, Dartmouth College)*, 2000.

[13] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, & M. Pinkerton, Cyberguide: a mobile context-aware tour guide, *Wireless Networks, 3*(5), 1997, 421 – 433.

[14] S. Meyer, & A. Rakotonirainy, A Survey of research on Context-Aware Homes, *Proc. Workshop Conference on Wearable, Invisible, Context-Aware, Ambient, Pervasive and Ubiquitous Computing,* Australia, *21*, 2003, 159 – 168.

[15] R. M. Gustavsen, Condor - an application framework for mobility-based context-aware applications. *Proc. UBICOMP 2002 Workshop on Concepts and Models for Ubiquitous Computing*, Goeteborg, Sweden, 2002.

[16] T. Strang, & C. Linnhoff-Popien, A context modeling survey, *The 1st International Workshop on Advanced Context Modeling, Reasoning and Management, Nottingham, 6th International Conference on Ubiquitous Computing (UbiComp1004),* UK, 2004, 33 – 40.

[17] H. Chen, T. Finin, & A. Joshi, An Ontology for Context-Aware Pervasive Computing Environments. IJCAI Workshop on Ontologies and Distributed Systems (Great Britain), 2003, 18, 3, 197 – 207.

[18] Zied Abid, Sophie Chabridon, Denis Conan, "A Framework for Quality of Context Management", Quality of Context, First International Workshop, QuaCon 2009, Stuttgart, Germany, June 25-26, 2009, pp. 120-131.

[19] Atif Manzoor, Hong-Linh Truong, Schahram Dustdar, "Quality Aware Context Information Aggregration System for Pervasive Environment", The 5th International Symposium on Web and Mobile Information Services (WAMIS 2009), (c) IEEE Computer Society, Bradford, UK, May 26-29, 2009.

[20] Hong-Linh Truong, Schahram Dustdar, "A Survey on Context-aware Web Service Systems", Invited Paper, International Journal of Web Information Systems, 5(1):5 - 31, (c) Emerald, 2009.

[21] M. J. Franklin, K. Tan, and C. Lui, Mobile Data Management. Go online to *http://books.google.mu/, 15.*

[22] J. Ensing, Software architecture for the support of context-aware applications, *Koninklijke Philips Electronics N.V. 2002*, UR 2002/841, 2002.

[23] A. K. Dey, & G. D. Abowd, The Context Toolkit: Aiding the development of context-aware applications, *Workshop on Software Engineering for Wearable and Pervasive Computing*, Limerick, Ireland, 2000.

[24] A. K. Dey, D. Salber, M. Futakawa, & G. Abowd, An architecture to support context-aware applications, *GVU Technical Report*, GIT-GVU-99-93, Georgia Institute of Technology, 1999.

[25] A. K. Dey, G. D. Abowd, & D. Salber, A context-based infrastructure for smart environments, *Proc. 1st International Workshop on Managing Interactions in Smart Environments (MANSE'99)*, Ireland, 1999, 114 – 128.

[26] P. Fahy, & S. Clarke, CASS - Middleware for mobile context-aware applications. *Proc. 2nd ACM SIGMOBILE International Conference on Mobile Systems, Applications and Services (Mobisys'2004)*, USA, 2004.

[27] T. Gu, H. K. Pung, & D. Q. Zhang, A Middleware for building context-aware mobile services. *Proc. IEEE Vehicular Technology Conference (VTC-Spring 2004)*, Italy, 2004.

[28] T. Gu, H. K Pung, & D. Q. Zhang, A service-oriented middleware for building context-aware services, *Journal of Network and Computer Applications, 28*(1), 2005, 1 – 18.

[29] A. Ranganathan, J. Muhtadi, S. Chetan, R. Campbell, & M. D. Mickunas, MiddleWhere: A Middleware for location awareness in ubiquitous computing Applications, *Proc. 5th ACM/IFIP/USENIX International Conference on Middleware*, Canada, 2004, 397 – 416.

[30] A. Ranganathan, & R. H. Campbell, A Middleware for context-aware agents in ubiquitous computing environments, *ACM/IFIP/USENIX International Middleware Conference 2003*, Germany, 2003, 143 – 161.

[31] O. Davidyuk, J. Riekki, V. Rautio, & J. Sun, Context-aware Middleware for mobile multimedia applications. *Proc. 3rd International Conference on Mobile and Ubiquitous Multimedia (MUM'04)*, Maryland, 83, 2004, 213 – 220.

[32] H. Naguib, G. Coulouris, & S. Mitchell, Middleware support for context-aware multimedia applications. *Proc. IFIP TC6/WG6.1 3rd International Working Conference on New Developments in Distributed Applications and Interoperable Systems*, Netherland, 198, 2001, 9 – 22.

[33] S. Khungar, & J. Riekki, A context based storage for ubiquitous computing applications. *Proc. 2nd European Union symposium on Ambient intelligence*, Netherlands, 84, 2004, 55 – 58.

[34] H. Chen, T. Finn, & A. Joshi, An ontology for context-aware pervasive computing environments, *IJCAI Workshop on Ontologies and Distributed Systems*, Great Britain, *18*(3), 2003, 197 – 207.

[35] O. Riva, Contory: A Middleware for the provisioning of context information on smart phones, *Proc. 7th ACM International Middleware Conference (Middleware'06), 4290*, Springer, 2006, 219 – 239.

[36] D. Sathan, A. Meetoo and R. K. Subramaniam, Context Aware Lightweight Energy Efficient Framework, WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY, 52, April 2009.

[37] C. G. Carlson, & D. E. Clay, The Earth Model - Calculating Field Size and Distances between Points using GPS Coordinates, *Site Specific Management Guidelines*, 2002.

[38] A. Meetoo & K. K. Khedo, SAIsense: A Novel Scalable, Adaptive and Intelligent Context-Aware Architecture, International Journal of Computers and Applications, 2011, 33 (3).

[39] D. L. McGuiness, & F. Harmelen, OWL Web Ontology Language Overview, W3C Recommendation February 2004. [Online] Available from: http://www.w3.org/TR/owl-features/ [Accessed 2011].

[40] M. K. Smith, C. Welty, R. Voltz, & D. McGuiness, OWL Web Ontology Language Guide, W3C Recommendation February 2004. [Online] Available from: http://www.w3.org/TR/owl-guide/ [Accessed 2011].

[41] Z. Zhang, & J. A. Miller, Ontology Query Languages for the Semantic Web: A Performance Evaluation, *Master Thesis (University of Georgia)*, 2005.

[42] G. Meditskos, & N. Bassiliades, Towards an Object-Oriented Reasoning System for OWL, *International Workshop on OWL Experiences and Directions, B. Cuenca Grau, I. Horrocks, B. Parsia, P. Patel-Schneider (Ed.)*, Ireland, 2005.

[43] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, & M. Dean, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission May 2004. [Online] Available at: http://www.w3.org/Submission/SWRL/ [Accessed 2011].

[44] A. Ranganathan & R. H. Campbell, A Middleware for Context-Aware Agents in Ubiquitous Computing Environments, In ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, Jun 16-20, 2003.

[45] M. C. O'Connor, FCC Certifies Ubisense's UWB, RFID Journal, The World's RFID Authority. [Online] Available from: http://www.rfidjournal.com [Accessed 2006].

[46] G. Roussos, Location Sensing Technologies and Applications, *School of Computer Science and Information Systems Birkbeck College, University of London*, 2002.

[47] R. Bridgelall, Characterization of Protocol-compatible Bluetooth/802.11 RFID Tags. [Online] Available from: http://rfdesign.com [Accessed 2006].