An Ontology-based Context Management System for Smart Environments

Laura M. McAvoy[1], Liming Chen[*], Mark Donnelly [*]
University of Ulster, UK
[1] McAvoy-L3@email.ulster.ac.uk
[*]{l.chen, mp.donnelly}@ulster.ac.uk

Abstract **– This paper proposes an ontology-enabled system for context management for smart environments. Central to the system is ontological sensor modelling, which attaches metadata and meaning to sensor data, thus supporting data repurposing and high-level content recognition. In addition, semantic sensor descriptions allow sensors to be automatically identified whenever they are put into an environment. Based on this, a novel plug-n-measure data acquisition mechanism has been developed to automatically detect and recognise new devices and update the contextual data relating to these devices on a real-time basis. The context management system has been developed based on the latest semantic technologies and deployed in an intelligent meeting room. The paper describes an experiment and presents initial results, which has demonstrated that the system is working.**

Keywords-context management; ontology-based context modelling; inference and reasoning; smart environments; plug-n-measure mechanism.

I. INTRODUCTION

In a smart environment, computer systems interact seamlessly on a continual basis with occupants through the use of context-aware enabling technology [1]. Context is a term used to define any information which can be used to characterise the state of an entity. An entity is defined as “a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves” [2]. Context-awareness is the ability to use this context in a relevant manner to update information, which may be of use to a user based on what they are doing at that time.

It is possible to make an environment smart and context aware by monitoring an inhabitant’s behaviours and environment in real-time along with other aspects. These other aspects include; fusing and interpreting the multiple modalities of signals and features, inferring the behaviour, anomalies or changes of the inhabitant continuously and providing application specific functions to the inhabitant’s needs and requirements [3]. Context is therefore very important within Smart Environments.

Various context management systems exist, including “The context toolkit” [4], the framework for managing context information within mobile devices [5] and the framework [6] which provides a graphical user interface (GUI) to the user.

However, whilst progress has been made within the area of smart environments and context management systems, some challenges still remain. This paper proposes an integrated context modelling and management system to deal with some of these challenges. The challenges are performing the repurposing of data [21], temporal reasoning [22] and multiple levels of context information [23]. The system which is proposed in this paper uses semantic technologies in particular, ontologies to model sensors and smart devices across a number of smart environment domains and exploit semantic reasoning to reason, store and share this information for various applications.

In previous work, the W4H model [23] is based on disaster specific domains and the context information associated with them. Unlike the W4H system which is domain specific, the system proposed in this paper can deal with context information and the multiple levels of it across a wide variety of domains through the use of a diverse ontology. The system is also able to handle large amounts of data which may be associated with certain larger domains due to the semantic repository that is used. Whilst temporal reasoning has been previously researched [22], the ability to perform temporal reasoning and access context histories remains a challenge. This system allows applications to access current context, temporal reasoning and context histories which may be of use to them. Finally the repurposing of data can be achieved through the use of the various architecture components.

A novel plug-n-measure mechanism for data acquisition is also introduced and discussed within this paper. The mechanism can detect new devices which are added to the environment, on a real-time basis. The context management system was also tested on a real-time basis within an intelligent meeting room as part of an experiment and application scenario. What the paper does not address is data extraction and analysis, which still has to be performed on the collected data.

The rest of this paper is organised as follows; Section II examines related work, opportunities and challenges within this area. Section III discusses the system architecture and the components that reside within. Ontology-based context management is described and a plug-n-measure mechanism is introduced and discussed. Section IV presents the system implementation and evaluation before discussing the experiment design within an intelligent meeting room. Data collection and the application scenario are also discussed in Section IV, before the paper concludes in Section V.

II. RELATED WORK

There are three leading approaches to context modelling. These are key-value [7], object-oriented [8] and ontology-based [9]. The current paper focuses on extending existing work in ontology-based context modelling. Both context modelling and context management have been areas of great interest within the research community, with a lot of work being done by many researchers within these areas.

Shih *et al.* [10] created a framework that used an upper level ontology to infer information about the user. The ontology used context deduced from sensor data and background information on the user, eliciting relationships between the two. Situation awareness was realised within the possible scenarios through the use of this ontology. First order logic rules were applied for inference and used alongside privacy aspects to deal with the privacy problem associated with situation awareness. The framework was aimed mainly at mobile devices and did not use real-time data.

A full context life cycle management was investigated by Jih *et al.* [11]. They used a context repository, ontology and reasoner to collect raw data, applying rules to the data and categorising the context information into one of three states; active, suspended and terminated. The raw data updates the context repository before being sent to the ontology to reason the data. Rules are then applied to infer about the data; two things happen from this point, a service is delivered and context inconsistencies are resolved before the data is sent to update the repository with the new consistent context data.

Ramparany *et al*. [12] used a central component within their work, a context manager. This manager collects context information from sensors, builds and maintains a situation model. As well as doing this, the manager can subscribe to any changes which are reported by each sensor which enables the updating or modification of the situation model. The sensors send information in the form of Resource Description Framework (RDF) [13] to the context manager.

Each research group creates and maintains a framework in a different way, usually aimed at the situation that is being dealt with and whether the information will be received on a real-time basis or not. All of them are aimed at storing and managing the information in a consistent manner.

Many opportunities and challenges exist within context modelling and management; the main opportunities which exist are within the scenario area, as a multitude of scenarios could be defined, using a context management system. Another opportunity however lies in the utilisation of applications from the deployment of the system. How other applications react can be extensive and it can range from updating mobile phone devices to creating an online booking system. The challenges which have been dealt with in the above work include the privacy problem associated with situation awareness [10], scalability and openness [12] and challenges associated with context i.e., information representation, context inconsistency and converting low level data into high level context [11]. The remaining challenges were discussed in Section I.

In this respect, this paper proposes a context modelling and management system for smart environments, alongside a novel context data acquisition mechanism. The system captures the sensor data, updates the repository and can be queried on a real-time basis by an application. The context data mechanism automatically detects new devices within the smart environment and updates the system accordingly. The ontology-based context modelling and management system which is proposed plans to overcome the challenges associated with context i.e., the repurposing of data, temporal reasoning and multiple levels of context information.

III. THE SYSTEM ARCHITECTURE

The ontology-enabled architecture (Fig. 1) illustrates the components which are required for this context modelling and management system. It consists of five main components; the devices component (contained within the Environment layer), sensor ontology component, enrichment component, semantic repository component and the API query and retrieval component. Each component relates to different parts of our system and environment; they are necessary to collect sensor data and change it from low-level into high-level context which can be re-used and shared.

The raw data comes from the devices and sensors which are located within the smart environment. This raw data is passed via the ontologies component to be stored as semantic data within the semantic repository. How this raw data is semantically enriched is discussed in more detail in Section IV C. The applications can access the semantic data which is stored in the semantic repository via the API query and retrieval component to update, retrieve or query the stored information. What is presented below is an outline of the main system components as illustrated in Fig. 1.

Devices component: This component deals with all sensors and devices which are deployed throughout the smart environment. When a sensor is activated, a signal is sent out to a receiver which is then displayed as data via a Java program running on a local computer. This data is saved into the context management system (CMS) automatically, both in the form of processed semantic metadata and raw data. The data which is outputted by the device component is low-level data, which can sometimes be ambiguous. By sending it to the CMS and passing it via the sensor ontology component, this ambiguity can be dealt with and corrected accordingly.

Semantic repository component: The semantic repository is where data is stored, both in raw and processed semantic data format. The data which has been outputted by the sensor device is passed to the enrichment component where it is made semantically rich. This enriched data gets saved to the semantic repository in the form of triples (subject, predicate and object). Metadata and meaning is added to the data within this component.

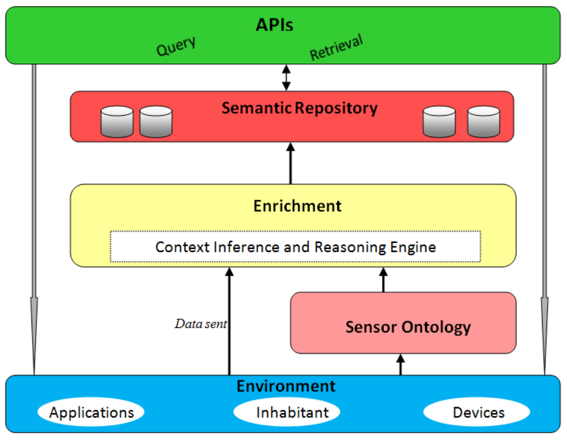


Figure 1. Architecture of the proposed context management system.

Sensor ontology component: This component is made up of ontology-based context models, which model the environment and the sensing devices within it. The context models can be applied to a wide variety of domains and devices. The ontologies are used like a schema to define the data as entities, individuals and properties. New devices can be added to the model using this component. The output from the sensor ontology and enrichment component is processed semantic data, which can be stored in the semantic repository and used by applications.

Enrichment component: The enrichment component consists of the context inference and reasoning engine, which is where the data gets sent after being passed to the sensor ontology component. Inference and reasoning rules are used against the data to process it and then the processed data is sent to the repository component.

Applications component: Applications use the processed data from the semantic repository through the use of the API component. This component uses the data queried or retrieved from the API component to update a user on a real-time basis.

API query and retrieval component: The API query and retrieval component provides facilities for the retrieval, query and updating of the information stored within the semantic repository.

All of these components placed together form the ontology-enabled system, which takes low level primitive sensor data and changes it into high-level context information. The CMS keeps the context information up to date and ensures that the information available to requesting applications is correct and relevant.

A. Sensor Modelling

The sensing devices within the smart environment have to be modelled to be semantically enriched. An ontology represents data in a formal manner by using entities and the relationships which link them together. Due to an ontology-based context model's use in knowledge sharing, reuse and reasoning, it is ideal for use within a smart research environment, various aspects such as multiple sensor data and heterogeneity can be dealt with through ontology-based context modelling. It is able to deal with context on all levels required within a smart environment as it can be divided into upper-level and lower-level ontologies.

By first creating a basic sensor ontology for the Smart Environments Research Group (SERG) domain, modelling the various sensors that exist along with their attributes the ontology could then be expanded upon. Once a basic sensor ontology had been designed using some of the devices from within the research environment it was then possible to expand the basic ontology by importing a larger, more diverse ontology which is discussed more in Section IV A. This ontology gave a wide overview of classes, subclasses and properties which could be applied to a wide variety of domains. By extending the original ontology with the developed sensor ontology [16], it was then possible to edit some of the properties and subclasses to meet desired specifications, an example of the sensor section can be seen in Fig. 2; this created an ontology which can be applied to multi-sensor domains which may include an array of smart devices. This extended ontology should now cover the majority of domains and devices within the smart environment world, with only a very small amount of future devices or domains needing to be created from the beginning. As well as being able to be used across a diverse set of domains, the extended ontology can also deal with multiple levels of context information found within smart environments.

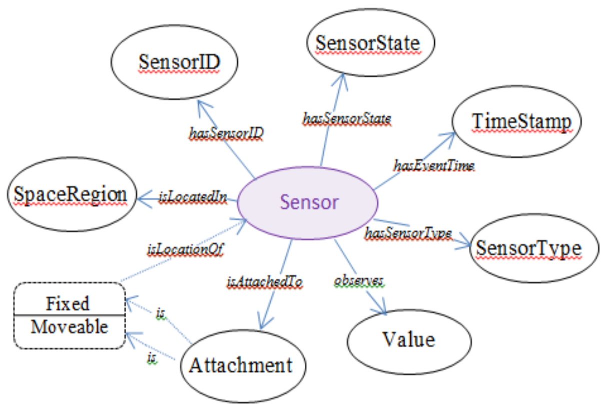


Figure 2. Example of the sensor ontology representation.

By using ontologies, it is possible to collect the data from the devices and add semantically rich metadata to this data. New devices can also be added in at this stage, ensuring that the ontologies are up to date with the devices that are in the environment. The ontology can be added directly to the semantic repository and used to semantically enrich the data which is added to the repository via a Java program. The reason that the environment is modelled in this way is that data can be ambiguous and ontology-based context modelling is one of the strongest types of context models available with the ability to deal with incomplete, ambiguous and informal data [14]. It ensures that contradictory information does not exist e.g., a device cannot be both a passive infra-red (PIR) sensor and a pressure sensor. Using this type of context model shall lower the amount of ambiguous data which is stored within the data repository. Ontology-based context modelling can also easily be applied to existing environments, which can be an advantage when it comes to sharing information across various domains and amongst numerous people.

The data which is obtained from the smart research environment is stored within the semantic repository, as previously discussed. This data can then be managed via the API component, using a query language. Finally it is difficult to use and distribute the same data amongst different applications and platforms due to the heterogeneous nature of different devices. By modelling the devices within the ontology, it makes it easier to disseminate across different platforms.

B. Plug-N-Measure Mechanism

A unique context data acquisition tool, which shall be referred to as a plug-n-measure mechanism throughout this paper, was created. A plug-n-measure mechanism as defined in this paper is similar to plug-n-play however the device does not have to be connected to the computer. A plug-n-measure mechanism can be defined as a context data acquisition tool which automatically detects new devices within a smart environment on a real-time basis. The mechanism then updates the system accordingly with the relevant context information pertaining to this new device.

Normally, when a new device is added to a smart environment, a person has to install the device, the drivers and manually update any information pertaining to that device. This can be a time-consuming effort, especially as the person may have to go through saved data and update information relating to an unknown device manually. As well as being time-consuming it can also lead to ambiguous or incomplete data being saved. By using alpha-numeric information associated with each device, the mechanism can add the device and information relating to the device to the system. The mechanism can also update any related information and finds the drivers, where possible, to install them automatically. By having the plug-n-measure mechanism in place, the likelihood of ambiguous or incomplete data is vastly lowered.

Using identifiers which are added to the incoming sensor data from the smart environment, the sensor type can be established. Other factors can then be used to determine where the sensor is located and what it is attached to. By modelling the sensor information within the ontology, semantic metadata pertaining to the new sensor can be added to the semantic repository. As new information about the sensor is determined, this can be updated within the repository in the form of semantic metadata. When new data is received from any device, the data is appended to previous information within the repository which relates to that device, saving anyone from having to go through context histories and manually update the sensor information for the new device.

IV. SYSTEM IMPLEMENTATION AND EVALUATION

An ontology-based context management system was developed. In order to validate the approach a controlled experiment was set up, data was collected and an application scenario was designed. The following sub-sections discuss the system implementation and use within the smart environment.

A. System Implementation

This section discusses how the system was implemented, which technologies were used in the implementation process and why.

The ontology-based context management system consists of a range of components some were newly created, i.e., the ontologies component which was further expanded by importing an existing ontology, whilst others were created or used from existing sources i.e., the semantic repository component, Sesame.

To create the ontology, Protégé [15] was used for the development environment. A basic sensor ontology was created within Protégé, depicting a wide range of sensing devices which may be available within a smart environment. A date and time entity was then added, along with other entities, such as possible locations within a smart environment. To further expand and enrich this basic ontology, the semantic sensor network (SSN) ontology [16] was imported enabling the use of the ontology across a wider number of domains. By using web ontology language (OWL) [17] and RDF [13], formal support for logical reasoning is provided along with expressivity and the ability to share information more readily amongst applications.

Once the ontology had been created, semantic repositories were researched, before deciding upon Sesame [18] for use within this system. Sesame was chosen due to the ability to work well with Java programming, which was essential to link in to the server side software. Sesame also offers the opportunity to use a number of different types of repositories, for this system a native RDF repository with RDF Schema inferencing was used. However, the system can also work as a native RDF repository without RDF Schema inferencing or as an in-memory RDF repository with or without RDF Schema inferencing. The semantic repository is one of the main components within this system and it was therefore necessary to have a repository which could store metadata, be updated on a real-time basis and be easy to query from an application. Sesame handles data in an easy to understand format and can be queried both from the Java program and from Sesame's own interface, workbench.

Once Sesame was added to the system, the ontology was further refined and sensors from the smart environment were updated and added to the ontology. For testing purposes the ontology was then uploaded to Sesame and triples were manually added to Sesame using their workbench interface. This ensured that the ontology worked alongside Sesame and any incoming data. After this had been established the Java program was connected to Sesame and run; real-time data was semantically enriched and stored in Sesame as triples.

Now that the main components within the system worked on a real-time basis, the functions component was assessed. Due to Sesame working well with SPARQL Protocol and RDF Query Language (SPARQL) [19], the ongoing research work with SPARQL and the fact this query language was developed for use with RDF, using this as the APIs component made sense. Diverse queries can be made using SPARQL and Sesame offers an API to use with this query language. By being able to use the API in the Java program, connecting the APIs component to the semantic repository was easy. The applications component which is programmed in Java could use SPARQL to query, retrieve and update data from Sesame.

Once all of the implementation had been completed, the next step was to test the system on a larger scale real-time basis, as part of an experiment.

B. Experimental Design

Based across one entire floor of a building, SERG utilises the smart research environment to undertake research which could assist people with their everyday lives [20]. The main devices are located within a dedicated smart kitchen, living-room and meeting room where experiments frequently take place. These devices are modelled in an ontology-based context model to make sense of the environment.

The meeting room contains a small table, 6 chairs, an interactive white board, a projector and a computer (Fig. 3). There are also a variety of sensors deployed within this room, which can help when implementing a scenario. The sensors which are of main interest are the six pressure sensors positioned on the chairs, the six accelerometers attached to each chair, the three PIR sensors on the walls, the three contact sensors on the doors and the audio sensors which detect the noise level within the room

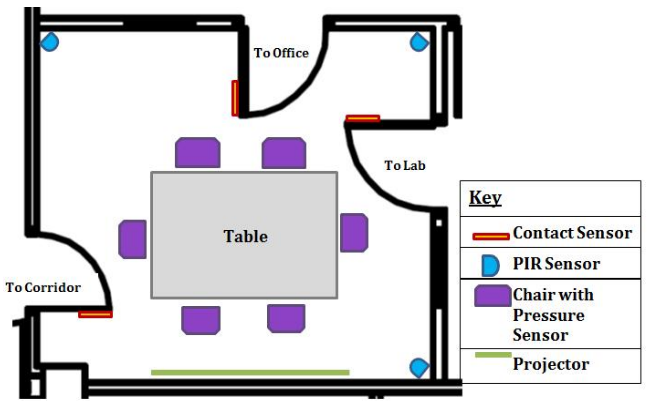


Figure 3. The meeting room layout.

without recording sound. The data from the pressure sensors can be used to determine when a person is sitting in the seat and when they have stood up. Along with the pressure sensors, the accelerometer devices are also attached individually to each chair, these devices determine when the chair is moved in or out from the table. Used together, all of these sensors can depict whether a participant is in the meeting room and if a meeting is taking place. All of this information is also saved as raw data and includes a timestamp to be cross-referenced against the semantic metadata at a later date.

C. Data Collection

When a sensor is triggered, numerical information is sent to an interface logger, which is connected to a computer. Identifiers are also added to the numerical information within the Java program, such as sensor status, sensor ID, sensor type and a timestamp. All of this information is modelled in the previously discussed ontology, and semantically enriched with additional metadata pertaining to each sensor. The sensor ID and/or the sensor type identifiers are matched with the correlating information within the ontology and additional metadata, which has been pre-modelled, is added to the raw data. The semantically enriched data is stored in Sesame in the form of triples (Fig. 4). This data can then be queried, retrieved and updated using SPARQL.

The sensor ontology resides within the semantic repository and incoming sensor information is appended to the ontological information relating to that sensor. A context identifier can also be added to the SPO triples to create subject, predicate, object, context (SPOC). The context referred to in this manner relates to context identifiers and if used can group related context information together, making it easier for a user to query a group.

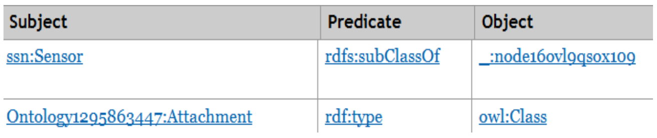
D. Application Scenario

Whether a meeting is taking place or when a meeting room is available for use can be established by using the sensors in the meeting room. If a meeting is taking place, the contact sensors on the doors should be activated, followed by at least two of the pressure sensors and accelerometers on the chairs being activated. It may also be possible to track numerous peoples' movement within the room through the use of passive infra-red (PIR) sensors; finally, audio sensors could pick up amplitude within the room. All of these sensor activations placed together form a scenario which suggests that a meeting is taking place. As each sensor is activated, the information is sent to the system and updated to processed semantic metadata, before being passed to the applications within the room.

To evaluate the system, it was run on a real-time basis. The running of the system was to establish if it worked correctly and also if it was possible to ascertain whether a user was passing through the meeting room, a meeting was taking place or one person was using it for a conference call. By querying the repository, which sensors were activated and in which order could also be established. The system was able to save the information on a real-time basis and it was possible to query and retrieve the information to determine what had happened within the meeting room. The raw sensor data had semantic metadata added to it and was saved accordingly within the system. The plug-n-measure mechanism could also detect when a new sensor had been added to the environment and was able to update the data accordingly. The application scenarios discussed in the rest of this section are only an example of what could be achieved.

Dr. Smith, Dr. Jones and Dr. Bloggs have arranged to have a meeting together. As they enter the intelligent meeting room, their mobile phones automatically switch to meeting mode. This is handled by the devices, enrichment, API and applications components within the CMS. As they are seated the LCD display outside the meeting room displays the message "meeting in progress". Detecting that Dr. Jones is in the meeting room via the devices and enrichment component, the CMS sets up the projector and computer accordingly, allowing Dr. Jones to make the presentation that he had previously noted down in his electronic calendar. The calendars for Dr. Smith, Dr. Jones and Dr. Bloggs change their statuses to "in a meeting" and update their appointments for the rest of the day accordingly via the applications and API components. If other users want to use the meeting room whilst it is occupied, they can query it via the API component and will be notified via mobile or email that the meeting room is occupied. They will also be updated on free timeslots that have not been pre-booked for the selected room.

Ms. Doe wishes to use the meeting room, but she does not want to pre-book it as she does not have a definitive meeting time. By using the CMS on an "as and when required basis" she can retrieve context information. Ms. Doe uses the application component to input her preferred room. The component infers from the CMS through the retrieval and reasoning on sensor data, that the room Ms. Doe wishes to book is in use. She receives this information, along with a selection of possibly free rooms and she repeats the same process via the CMS to find a vacant room. Once she finds a vacant room, she can use it and the CMS is updated with relevant information to notify other users that, that room is now occupied.



Subject	Predicate	Object
ssn:Sensor	rdfs:subClassOf	_:node16ovl9qsox109
Ontology1295863447:Attachment	rdf:type	owl:Class

Figure 4. Triples: as stored within the Semantic Repository.

Another way in which the CMS can be used is by management personnel. Using the applications component, they query and retrieve the stored data which can then be visualised using a visualisation tool. Based on this information, management can see how each room is used and set up additional rooms accordingly, e.g., a room specifically for conference calls etc.

V. CONCLUSION AND FUTURE WORK

The study presented in this paper addressed the challenges with regard to the ambiguity of collected data, temporal reasoning and sharing and re-using data amongst various applications. Temporal reasoning is also a problem as many application scenarios within smart environments closely relate to a temporal sequence of events. These events usually form a situation which is modelled via the low-level sensor data which can be collected from a smart environment; not only do applications want to access the current information they may also want to access context histories. These context histories usually contain the high-level information.

An ontology-based context management system was created along with an adaptive contextual data acquisition mechanism for use within a smart environment. Through the implementation and testing of this system on a real-time basis within a meeting room scenario, results were obtained which showed that low-level sensor data could be captured and semantically enriched with metadata and stored within a semantic repository. The plug-n-measure mechanism was able to recognise new devices which were deployed within the environment and update the system with contextual data. By adding semantic metadata to low-level sensor data, it makes it easier to disseminate the high level context across multiple domains and platforms. The semantic repository and functions component of the system make it possible for applications to access the current information as well as context histories.

The study has shown that the presented context management system can be applied to numerous scenarios and leads to an ample amount of opportunities for proposed applications. Low-level sensor data can be semantically enriched to high-level context enabling the sharing, re-use and reasoning of the context by different applications, platforms and domains.

The implementation of the system shows that the system can recognise sensors and other devices as they are added to the smart environment. It enables the repurposing of contextual data as well as a high level context inference based on temporal reasoning and domain heuristics. The future work includes analysis and extraction of the meeting room data for high level context use, i.e., activity recognition within the smart environment.

REFERENCES

[1] D. J. Cook and S. K. Das, “How smart are our environments? An updated look at the state of the art”, *Pervasive and Mobile Computing*, vol. 3, pp. 53–73, 2007.

[2] A. K. Dey, "Understanding and Using Context," vol. Personal and Ubiquitous Computing (2001), pp. 4-7, 2001.

[3] L. Chen, C. D. Nugent, and H. Wang, “A Knowledge-Driven Approach to Activity Recognition in Smart Homes”, *IEEE transactions on knowledge and data engineering*, vol. 24, no. 6, pp. 961-974, 2010.

[4] A.K. Dey, G. D. Abowd, and D. Salber, “A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications', *Human-Computer Interaction*, vol. 16, no. 2, pp. 97-166, 2009.

[5] P. Korpipää, E.Malm, I. Salminen, T. Rantakokko, V. Kyllönen, and I. Känsälä, “Context Management for End user Development of Context-Aware Applications”, MDM 2005, Ayia Napa Cyprus, pp. 304-308, 2005.

[6] H. van Kranenburg, M. S. Bargh, S. Lacob, and A. Peddemors, “A context management framework for supporting context-aware distributed applications”, *IEEE Communications Magazine*, August 2006, pp. 67-74, 2006.

[7] M. Samulowitz, F. Michahelles, and C. Linnhoff-Popien, "CAPEUS: An architecture for context-aware selection and execution of services." in *Third International Working Conference on New Developments in Distributed Applications and Interoperable Systems*, The Netherlands, pp. 23-40, 2001.

[8] K. Cheverst, K. Mitchell, and N. Davies, "Design of an object model for a context sensitive tourist GUIDE." vol. Computers and Graphics 23, pp. 883-891, 1999.

[9] H. Chen, T. Finin, and A. Joshi, “Using OWL in a Pervasive Computing Broker”, In Proceedings of Workshop on Ontologies in Open Agent Systems (AAMAS’03), pp. 9-16, 2003.

[10] F. Shih, V. Narayanana, and L. Kuhn, “Enabling Semantic Understanding of Situations from Contextual Data in a Privacy-Sensitive Manner”, Activity Context Representation – Techniques and Languages: Papers from the 2011 AAAI Workshop (WS-11-04), pp. 68-73, 2011.

[11] W. Jih, C. Huang, and J. Hsu, “Context Life Cycle Management in Smart Space Environments”, Proceedings of the 3rd workshop on Agent-oriented software engineering challenges for ubiquitous and pervasive computing (*AUPC’09)*, London, UK, pp. 9-14, 2009.

[12] F. Ramparany, Y. Benazzouz, L. Chotard, and E. Coly, “Context Aware Assistant for the Aging and Dependent Society”, Workshop Proceedings of the 7th International Conference on Intelligent Environments 2011, Nottingham, UK, pp. 798-809, 2011.

[13] RDF – Semantic Web Standardds, “Resource Description Framework (RDF)”, W3C 2004 [Online], Available: http://www.w3.org/RDF/ [retrieved: July, 2012].

[14] T. Strang and C. Linnhoff-Popien, "A context modeling survey", Proc. First International Workshop of Advanced Context Modelling, Reasoning and Management at UbiComp 2004, Nottingham, UK, 2004, pp. 33-40.

[15] The Protégé Ontology Editor and Knowledge Acquisition System, “Welcome to Protégé”, 2012 [Online], Available: http://protege.stanford.edu [retrieved: July, 2012].

[16] W3C Semantic Sensor Network Incubator Group, “*Incubator Activity > W3C Semantic Sensor Network Incubator Group*”, W3C 2011 [Online], Available: http://www.w3.org/2005/Incubator/ssn/ [retrieved: July, 2012].

[17] OWL - Semantic Web Standards, “*Web Ontology Language (OWL)*”, 2010 [Online], Available: http://www.w3.org/2001/sw/wiki/OWL [retrieved: July, 2012].

[18] OpenRDF.org: Home, “openRDF.org … home of Sesame”, 2012 [Online], Available: http://openrdf.org [retrieved: July, 2012].

[19] SPARQL Query Language for RDF, “SPARQL Query Language for RDF”, 2008, Available: http://www.w3.org/TR/rdf-sparql-query/ [retrieved: July, 2012].

[20] C.D. Nugent, M.D. Mulvenna, X. Hong, and S. Devlin, “Experiences in the development of a smart lab”, *The International Journal of Biomedical Engineering and Technology* (IJBET ’09), vol. 2, no.4, pp.319-331, 2009.

[21] L. Chen, C. D. Nugent, M. Mulvenna, D. Finlay, and X. Hong, "Semantic smart homes: Towards knowledge rich assisted living environments," in Springer Berlin/Heidelberg, 2009, pp. 279-296.

[22] L. Hsien-Chou and T. Chien-Chih, "A RDF and OWL-Based Temporal Context Reasoning Model for Smart Home," *Information Technology Journal*, vol. 6, pp. 1130-1138, 2007.

[23] H. Truong, A. Manzoor, and S. Dustdar, "On modeling, collecting and utilizing context information for disaster responses in pervasive environments" 24th August 2009, vol. CASTA '09, pp. 25-28, 2009.