*Journal of Universal Computer Science, vol. 12, no. 3 (2006), 357-366*
*submitted: 30/10/05, accepted: 15/1/06, appeared: 28/3/06 © J.UCS*

# Information and Hybrid Architecture Model of the OCP Contextual Information Management System

**Ignacio Nieto**
(University of Murcia, Spain
inieto@um.es)

**Juan A. Botía**
(University of Murcia, Spain
juanbot@um.es)

**Antonio F. Gómez-Skarmeta**
(University of Murcia, Spain
skarmeta@dif.um.es)

**Abstract:** This paper describes OCP (*Open Context Platform*), a middleware which provides support for management of contextual information and merging of information from different sources. The host system is thus endowed with proactive capacities which, in turn, provide a certain environmental intelligence [8]. The approach consists of a modelling of contextual information which is based on Semantic Web derived technologies and a description of the structured merging in the form of decision rules. The latter serve to classify situations of interest and subsequently to trigger off the relevant actions at each moment. Elsewhere, the underlying architecture in OCP is designed so that it can function both by centralizing all the contextual information from a central server and by distributing it among consumers and producers of this type of information.

**Keywords:** Context-awareness, semantic web, decision rules, hybrid architecture, information merging

**Categories:** C.2.4, H.3.3, I.2.11, J.4, K.4.2

## 1 Introduction

The need for software environments which adapt ever better to the demands of the users and their environment lead us towards context aware systems. Such systems handle all the relevant information that surrounds and is part of the users' environment. The preferences of the user, their likes, their location, their state of mind, their activity, their environment, the temperature of the room, lighting conditions etc., is all information which can be classified under the heading of *Contextual Information*. Hence, a context may store information on communication capacities, by storing and processing users' PDA, or it may contain data on a user's preferences for information display. The current trend, given the ever greater number of small devices and laptops, is towards distribution through heterogeneous systems made up of large numbers of networks of different characteristics, together with a set of users who have a range of devices of various natures like PDAs, laptops, desktop

PCs and even sensors, cameras or intelligent household appliances. A variety of challenges for constructing a contextual system arise from these scenarios. First we must be able to guarantee that the contextual information be treated in a heterogeneous way and in a format which allows transmission of the semantic characteristics proper to the context (see section 2). We then need to take into account the dynamic component of the information in the sense that it is constantly changing, and both the changes in the information and the subsequent notifications need to be handled effectively. Finally, there is the question of being able to guarantee that the contextual information is available at all times (see section 3).

The rest of this paper is structured as follows: section 2 deals with how OCP represents and manages contextual information. Section 3 describes the hybrid system we propose for context management, along with an analysis of its behaviour in various scenarios. Finally, section 4 gives some conclusions and indicates some of the lines of work we are currently involved in.

## 2 Representation of contextual information in OCP

The problem of representing context is approached nowadays as an open topic and as a new area of research. In general, and based on context, efforts have been devoted to defining both the concept of context – context aware systems and even the life cycle such systems should follow [6] - and the methods and mechanisms for capturing or obtaining contextual information [12]. Brown [2] and Finney [7] concur in stating that the use of context is closely bound up with the way in which we perceive it and that a correct interpretation is required if it is to be of use. Such an observation points us towards using ontologies to provide a common definition, with its semantics, of the information model that all parties involved in the application should use. The use of ontologies ensures a common framework within which context information can be easily interchanged, previously made context models can be reused, semantic interpretation can be made of the information in the form of knowledge and reasoning can be carried out on the context in order to integrate entities with a low level of information into other more abstract entities. Examples of ontological based systems for modelling can be seen in [13, 10]. The basic functioning resides in the use of technologies derived from OWL. It is clear that context modeling is following the same trend- The first study to use OWL to describe the information elements of a context aware system was based on the context broker CoBrA [3]. Within that study a first ontological standard was proposed, named Cobra-ont. This ontology reuses vocabulary from another ontology, SOUPA [4], related to ubiquitous computation, in which the management of contextual information is situated. SOUPA, in turn, reuses other standard ontologies like DAML-time, among others. Naturally, there is already in CoBrA an element which reasons over contextual information by making use of the advantages the use of ontologies offers. It is a deductive type of reasoning and is also used to check the wholeness of the information acquired with respect to that stored in the knowledge database. However, explicit rules are not defined for identifying events of interest, unlike the case previously cited [13].

## 2.1 Characteristics of OCP ontology

The philosophy behind OCP modelling includes reusability. When modelling people mechanisms, environments it does not make sense to model starting from nothing when there are numerous more or less standardized ontologies for each of the topics. We have taken this into account in OCP and in this sense our paper is similar to those by SOUPA [4] and Smart Spaces [14].

Today there are several ontologies available for context modelling, such as CONON [13], COBRA-ONT [3] or ULCO [14]. Of all these, the only one with reusability, which is desirable in all ontology, is COBRA-ONT, which has been extended to SOUPA. CONON, ULCO and COBRA-ONT use OWL as the base language. The availability of SOUPA and its level of description make it ideal for incorporation in our work. Figure 1 shows the focus of the paper. It should be observed that RDF is the base. Powerful data bases are being developed for this language which are aimed at tags and tuples which this language imposes to specify subject, object and predicate. On RDF, although not exhaustively (we must allow for the possibility of integrating languages directly based on RDF), there is RDF-S, which includes more advanced mechanisms for handling ontologies. We place OWL on RDF-S, and other languages based or not based RDF-S. SOUPA sits directly on OWL and stems from the OCP extensions to form the ontology of the context platform we propose.

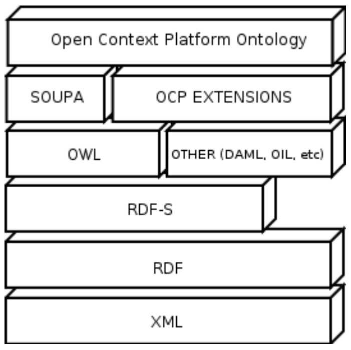
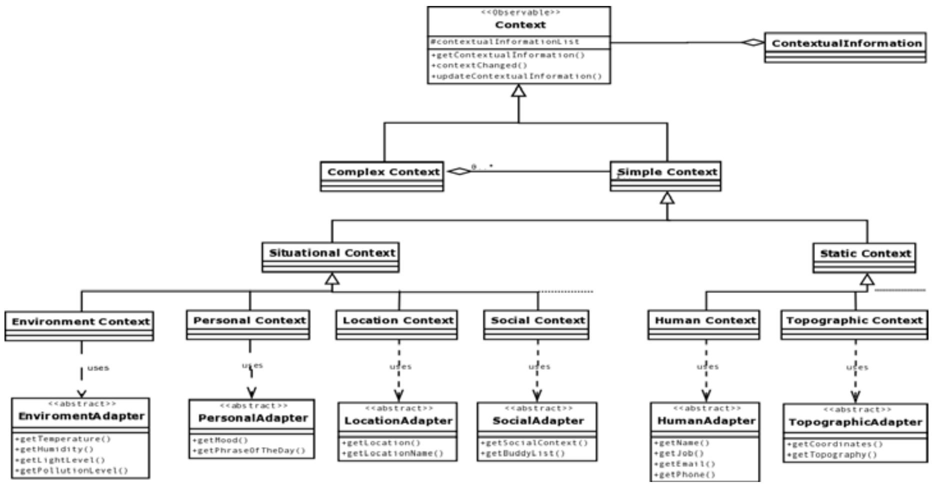


*Figure 1: Organization of ontologies by levels in OCP*

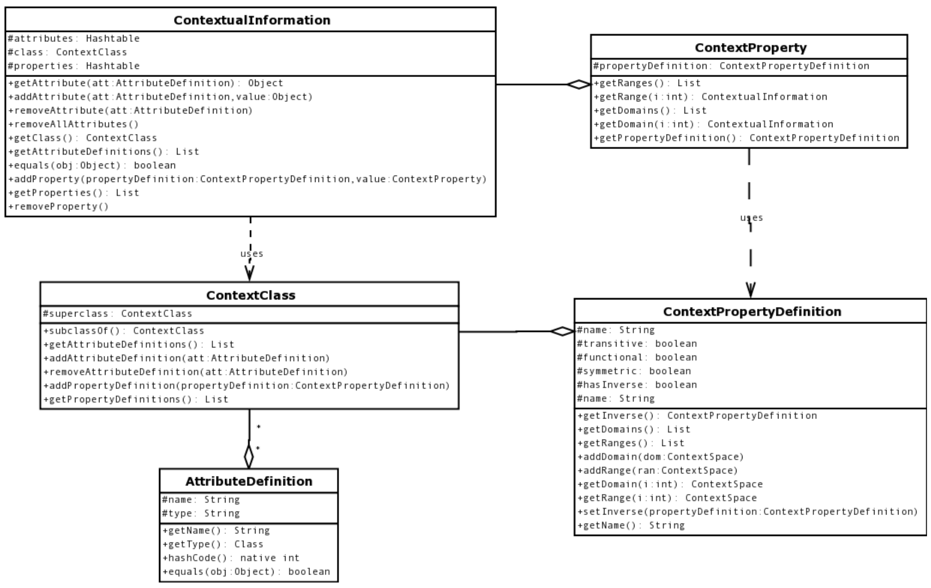
OCP has a generic ontology designed with the following aims in mind. In the first place is the separate consideration in the model of the changing and the invariable information in the model. To do this, we assume a dichotomic nature for the contextual information. On the one hand, there is information which remains static over time but which is nevertheless necessary for describing situations. On the other hand, there is information which changes dynamically and which from the point of view of infrastructure has to be considered in a special way. The first type of contextual information is known as *static context* and the second as *situation context*, as shown in figure 2. The second aim is to define a generic outline which is able to integrate previously defined ontologies for reuse (e.g. SOUPA) using OWL. Figure 2 shows that the class `ContextualInformation` corresponds directly to the concept of class instance `owl:class`. Observe that `ContextClass` corresponds to `owl:class`. As regards the properties of a class, we have those which relate a class instance with an instance of a basic type of data and those which relate a class instance with another class instance. The former are defined through `AttributeDefinition`, equivalent to `owl:DataTypeProperty` while the latter are defined through `ContextPropertyDefinition`, equivalent to `owl:ObjectProperty`. Elsewhere, the class `ContextSpace` is necessary to apply cardinality constraints (i.e. `Cardinality`) and membership ones (`Restriction`) to the properties.

This direct correspondence with OWL provides two important advantages. On the one hand there is the ease of integration of ontologies defined through this language and, on the other, there is easy maintenance for a possible future change to a language based on descriptive language [1].

Figure 2 shows the context hierarchy present in OCP. A context includes all the information which describes a specific situation, including people, mechanisms and other objects. The description of a certain situation is made up of a set of basic informations which are in some justified way incorporated, since in the majority of applications in real life we are handling complex contexts. Complex contexts are made up of several simple contexts, but which form a semantic unit with contextual coherence. This implies that if any element is eliminated from the complex context, the entity thus referred to remains incomplete since there is no information available that is important for decision taking. The concept of complex context is analogous to the context aggregator used in the Toolkit context [5]. The aim here is the development of simpler applications through minimizing the number of interactions of the applications with contextual entities to the extent that a single entity (i.e. a complex context) is always accessed.



(a)


ContextualInformation
#attributes: Hashtable
#class: ContextClass
#properties: Hashtable
+getAttribute(att:AttributeDefinition): Object
+addAttribute(att:AttributeDefinition,value:Object)
+removeAttribute(att:AttributeDefinition)
+removeAllAttributes()
+getClass(): ContextClass
+getAttributeDefinitions(): List
+equals(obj:Object): boolean
+addProperty(propertyDefinition:ContextPropertyDefinition,value:ContextProperty)
+getProperties(): List
+removeProperty()
ContextProperty
#propertyDefinition: ContextPropertyDefinition
+getRanges(): List
+getRange(i:int): ContextualInformation
+getDomains(): List
+getDomain(i:int): ContextualInformation
+getPropertyDefinition(): ContextPropertyDefinition
uses
uses
ContextClass
#superclass: ContextClass
+subclassOf(): ContextClass
+getAttributeDefinitions(): List
+addAttributeDefinition(att:AttributeDefinition)
+removeAttributeDefinition(att:AttributeDefinition)
+addPropertyDefinition(propertyDefinition:ContextPropertyDefinition)
+getPropertyDefinitions(): List
ContextPropertyDefinition
#name: String
#transitive: boolean
#functional: boolean
#symmetric: boolean
#hasInverse: boolean
#name: String
+getInverse(): ContextPropertyDefinition
+getDomains(): List
+getRanges(): List
+addDomain(dom:ContextSpace)
+addRange(ran:ContextSpace)
+getDomain(i:int): ContextSpace
+getRange(i:int): ContextSpace
+setInverse(propertyDefinition:ContextPropertyDefinition)
+getName(): String
*
*
AttributeDefinition
#name: String
#type: String
+getName(): String
+getType(): Class
+hashCode(): native int
+equals(obj:Object): boolean

*(b)*

*Figure 2: Context ontology in OCP: context hierarchy (a) and internal structure (b)*

## 2.2 Handling contextual information

Modelling ontology in OCP with OWL as a reference allows us to handle the semantics of the information. Our aim is to create a system capable of reacting proactively to the environment in which it is deployed and to its users. To accomplish this, the system must be able to interpret the meaning of the information which it handles, and to relate it to other contexts and information received from different sources, and thus a semantic representation of the information is required. The next step is to establish mechanisms for the correct handling of such semantic information. OWL only establishes the structure and framework the information must present in order to handle its semantic load, with no indication as to how the information is handled. This information is dealt with in OCP at the context level thanks to the *ContextHandlers*. A contexthandler is an entity which knows how to perform a certain specific functionality with a context, extracting and making use of the semantic load which it carries.

### 2.2.1 Context merging

The ContextMergers are in charge of mixing the contents of two or more contexts, thus generating a new context coherent with the information of them all, and one whose information is a mergence of all the contextual informations and is understood as a combination rather than a mere union of sets.

To see how contextmergers work, let us imagine the scenario of a Faculty with its lecturers and students in which an environmental intelligence system is installed which facilitates their daily tasks. Let us suppose a user about whom contextual information is received in OCP. On the one hand, the information states that the user is a lecturer. On the other hand, the localization module informs that the lecturer is in his office. If the hourly information coincides with the work schedule of the lecturer we can integrate this information to create a work context in which we indicate that the lecturer is in his office and is, moreover, available for tutorials. This integration is made through a rule-based inference motor, thanks to the above *ContextMergers*. These mergers possess specific rules to merge certain contexts according to the needs of the scenario in which the system is to be deployed. With these rules we can define how the contexts are merged according to our needs. The new contextual information generated contains a greater semantic load than the sum of the different parts from which it was generated. Moreover, we will be able to make certain adjustments by gathering together contextual information according to the selection criteria for the information we have already obtained, e.g. : Is the lecturer's current interface (a PDA) less comfortable than a switched off desk top ? If this is the case, can we switch on the PC and advise the lecturer that it can be used? Is the temperature suitable? If not, we can switch on the conditioning system in the lecturer's office? We can also integrate outstanding information on questions or messages directed to the lecturer which have not been received because he was elsewhere (e.g. at home, with friends, nit available, etc.) and which should be sent to him.

### 2.2.2 Automatic reasoning on contextual information

The context inference mechanism (ContextInference), which appears depicted in Figure 3, contains the intelligent action load and the decision making of the OCP system. Using a context inference it is possible to take a decision (and carry out an associated action) about specific contextual information. In OCP, an inference is made using a system of rules, so we can say that a context inference is made up of an association of rules, which may be of various types, (do-if rules, do-for-all rules, etc...). The result of an inference is a specific decision, which is translated into an action. This action may be simply a textual description of the action to be performed or it may include remote calling of a service or a modification of the contextual information being handled.

Just as the OCP contextual information follows the OWL patterns, so is the inference in OCP (which we have already said is performed through rules systems) carried out using the SWRL guidelines. SWRL (Semantic Web Rule Language) [11] is a language of rules for the a web. The advantage of SWRL, apart from the degree of standardization and abstraction which it provides, is that it offers many support tools, e.g. SWRL-Protégé Plugin,[1] which, at the same time as defining the ontology of contextual information, allow us to introduce the semantics and functionality of the rules systems into the ontologies and reflect them in manageable information that can be recovered, used and stored. With respect to the inference motor, OCP makes use of the Jena platform. Jena[2] is a java environment for the construction of semantic web applications. It provides a programming environment for RDF, RDFS and OWL, and includes a rules-based inference motor, which OCP makes use of. The Jena inference

1 http://protege.stanford.edu
2 http://www.hpl.hp.com/semweb/jena2.htm

subsystem was designed to be integrated or combined with inference motors or external reasoning systems.

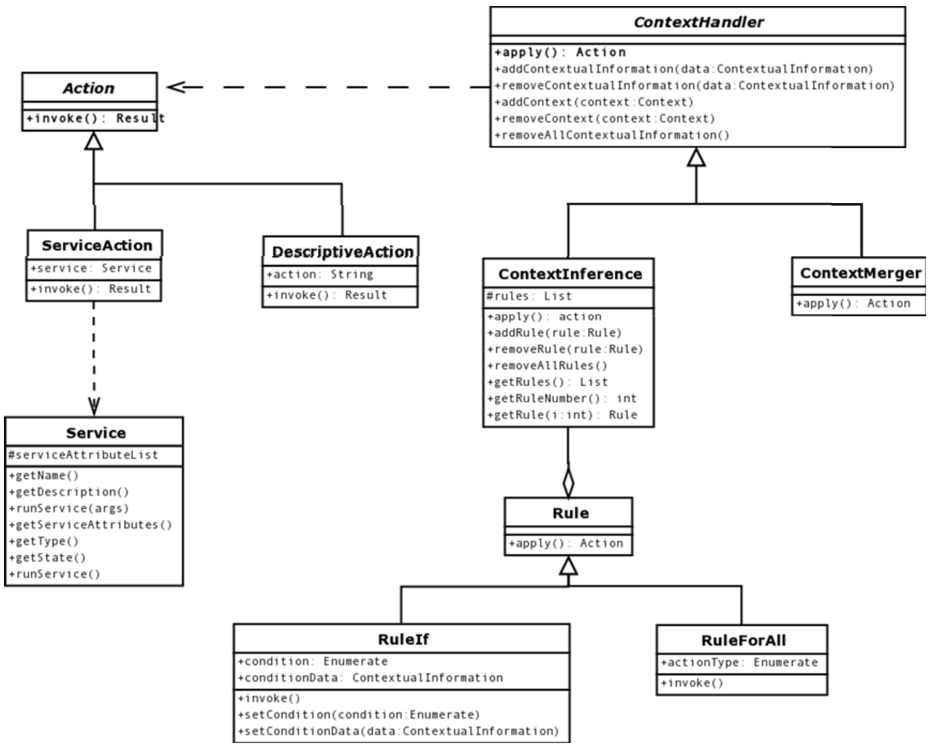


*Figure 3: Handling contextual information in OCP*

## 3 A hybrid system of contextual management

There are two main approaches to when constructing a system of information interchange among a group of users: a centralized approach or a distributed approach. Both have their advantages and drawbacks. A centralized architecture offers the guarantee that there is always a reliable context source made up of a series of nodes which form a “context centralized service” which users can refer to at any time to obtain quality, reliable contextual information. However, a centralized system has an inherent critical point whose availability cannot always be guaranteed in real systems. In contrast to these systems, the distributed ones offer adaptation and non dependence on a central control, although they do have certain disadvantages such as a limited processing capacity with the subsequent loss of reliability in functioning. Our interest lies in constructing a system that has the best of both approaches, by switching functioning modes as the occasion arises. As regards the application of this type of system, we are looking at the field of emergency systems. This type of system should be available following disasters (e.g. fires, earthquakes) to help out with rescue related tasks (e.g. evacuation of a building).

To construct a distributed contextual information system we require a structure that can store and manage information in a totally decentralized way. The first studies in this sphere were focused on the coordination paradigm which, based on the separation of the aspects of computation and the interaction of components integrating a system, gave rise to various models and languages which have been successfully applied in the development of parallel and distributed applications. The first coordination model developed was by Linda[9]. Linda introduces the concept of a tuple space. A tuple space is a space with associative memory in which the objects are referenced by content instead of by their address. These spaces function like logically distributed memory systems where users can write, read and delete information using a reduced set of simple primitives, such as `in(t)` to obtain information from the tuple space, `out(t)` to insert information in the tuple space, and `rd(t)` to read information without extracting it from the tuple space. Most commercial tuple spaces are based on the model described by Linda, e.g. JavaSpaces, TSpaces and GigaSpaces, and the model has also been the inspiration for the tuple spaces we propose for our system.

### 3.1 Context suppliers and tuple spaces

The tuple spaces used in our system are based on the existence of a distributed and shared storage space where entities can insert, obtain and even delete information. From the entities' viewpoint, the tuple space functions like an information blackboard, distributed throughout the different addresses. In our case, the tuple spaced is constructed with the local contextual information from each autonomous node in the system, and we call these CPP (*Context Provider Peer*). Each CPP can access this space to share its local data or to obtain data from other CPPs. The tuple space acts as a container where information has no fixed source, since it is presented in the form of information tuples which are defined by the information container. It thus constitutes a non reliable space of ever available context, as long as there is at least one user in the system. This makes it valuable in situations where no other contextual source can be found. Each CPP can always rely on its own contextual information (i.e. that which it produces itself).

After all this, we still require entities capable of guaranteeing a more reliable handling of contextual information. This is achieved through an additional element, the SCPP (*Super Context Provider Peer*). A SCPP serves contrasted, reliable contextual information. It is especially configured to gather, mix and modify this type of information and gives rise to a series of valuable contexts which CPPs can use. Each CPP can go to a SCPP to obtain contextual information about the environment, other users and even about external entities in the area surrounding the SCPP. Thus, we can state that a SCPP offers a centralized service of contextual information. It compares, shares and obtains new information on all the area covered by the system. The SCPP obtains this information from the CPPs (mobile mechanisms, environment sensors, applications or any other entity or source devoted to such ends).

## 4 Conclusions and future research

This article presents OCP, a hybrid system of context management which, by adapting to circumstances and events in the environment, is able to create a context aware, multi-user, heterogeneous system capable of switching between centralized and distributed behaviour as necessary in order to provide users with a reliable context source to which they can refer at all times. OCP seeks to have an ever ready complete contextual structure thanks to an ontology which it can make available to users and with which new, useful information can be obtained, be it through inference and comparison of information or through cooperation by individuals and data merging. The OCP system comes in the form of a set of standard classes, i.e. an API and, consequently, there is no single way of using them. In this sense, the information model is unified but not the integration of OCP in ubiquitous computing applications. The OCP system is currently being built on a tuples space infrastructure of communications. Is being tested in a number of ambient intelligence related research projects as the infrastructure for contextual information management.

The scenario we give presents many open questions and subsequent frameworks for development. OCP offers a framework for global development for context aware applications, and is thus useful in environmental intelligence systems and ubiquitous computation. The aim of OCP is for such systems to have a reference framework for managing contextual information so that they need only be concerned with the architecture and structure of the system itself, i.e. the framework for context management and interchange can be taken as developed. Elsewhere, when the OCP system starts to function in a cooperative distributed model, a while range of questions arises. For example, management of historic data is not deeply studied. It actually can be stored and recovered but not automatically used in decision processes. Isolated users have been left without any reliable source of information, and they know they are in a hostile environment where they cannot obtain quality data. They may seek support from the last information obtained from reliable context sources, but for how long? And how reliable is the information with the passing of time and with data becoming obsolete? If users obtain information from the tuple space, should they trust new information from another source? More so than the previous instructions from the last communication with the SCPP? And for how long? An important area for research and analysis is the definition of precise reliable rules and the length of the validity of information. Likewise important is the design of a reasoning system which can lead us to a correct, coherent and suitable conduct for these situations.

**Acknowledgements**

This work has been partially funded by the Spanish local CARM government through the Seneca Foundation and the Research Project ENCUENTRO 00511/PI/04.

## References

[1] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *Description Logic Handbook*, chapter An Introduction to Description Logics. Cambridge University Press, 2002.

[2] P. J. Brown. The stick-e document: A framework for creating context-aware applications. *Proceedings of the Electronic Publishing*, pages 259–272, 1996.

[3] H. Chen, T. Finin, and A. Joshi. An ontology for contextaware pervasive computing environments. *Knowledge Engineering Review. Special Issue on Ontologies for Distributed Systems*, 2003.

[4] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *International Conference on Mobile and Ubiquitous Systems: Networking and Services*, August 2004.

[5] Anind K. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.

[6] R. Want B. N. Schilit N. Adams R. Gold K. Petersen D. Greenberg J. Ellis and M. Weiser. An overview of the parctab ubiquitous computing environment. *IEEE Personal Communications, 2(6)*, pages 28–43, 1995.

[7] J. Finney and N. Davies. The flexible ubiquitous monitor project. *Proceedings of the Third Computer Networks Symposium*, 1996.

[8] W.A IJsselsteijn G. Riva, F. Davide, editor. *Being There: Concepts, effects and measurement of user presence in synthetic environments*. IOS Press, Amsterdam, The Netherlands, 2003.

[9] D. Gelernter. Generative communication in linda. *ACM Transactions on Programming Languages and Systems. Vol. 7, No. 1*, pages 80–112., 1985.

[10] Tao Gu, Hung Keng Pung, and Da Qing Zhang. Toward an OSGI-based infrastructure for context-aware applications. *IEEE PERVASIVE Computing*, pages 66–74, October-December 2004.

[11] Craig F. Smith H. Peter Alesso. *Developing Semantic Web Services*. A K Peters, Ltd., 2004.

[12] J. Pascoe N. S. Ryan and D. R. Morse. Human-computer-giraffe interaction - HCI in the field. *Workshop on Human Computer Interaction with Mobile Devices*, 1998.

[13] X. Wang. Ontology-based context modeling and reasoning using owl. In *Context Modeling and Reasoning Workshop at PerCom 2004.*, 2004.

[14] Xiaohang Wang, Jin Song Dong, Chung Yau Chin, Sanka Ravipriya Hettiarachchi, and Daqing Zhang. Semantic space: An infrastructure for smart spaces. *IEEE PERVASIVE computing*, 4:32–39, 2004.