An ontology-based context management and reasoning process for UbiComp applications

Eleni Christopoulou[1], Christos Goumopoulos[1] & Achilles Kameas[1], [2]

[1] Research Academic Computer Technology Institute, Research Unit 3, Design of Ambient Information Systems Group, N. Kazantzaki str., Rio Campus, 26500, Patras, Greece
{hristope, goumop, kameas}@cti.gr

[2] Hellenic Open University, School of Science & Technology, 23 Sahtouri str. 26222, Patras, Greece

Abstract

UbiComp applications operate within an extremely dynamic and heterogeneous environment and have to dynamically adapt to changes in their environment as a result of users' or other actors' activities. So context definition, representation, management and use become important factors that affect their operation. To ease the development of such applications it is necessary to decouple application composition from context acquisition and representation, and at the same time provide universal models and mechanisms to manage context. In this paper is presented an approach for building a context-aware UbiComp system organised in hierarchical levels. The focus of the paper is on an ontology-based context modelling, management and reasoning process developed for composing context-aware UbiComp applications from AmI artefacts.

1. Introduction

The human communication model represents an efficient and effective way to convey information, thoughts, feelings, etc. from person to person. This is based on the fact that in a social dialogue members are sharing an implicit awareness of related situations, of adjacent background, of context environment, and possess the skills to assess, infer and adapt their behaviour appropriately. However, in a human-computer interaction, such an implicit channel of understanding is missing.

Ambient Intelligence (AmI) evangelises computing so integrated into everyday objects that it becomes invisible to users [19]. Hidden computers and ubiquitous computing (UbiComp) applications should take into account the user and environmental context and adjust their behaviour in order to improve the provided services to humans [20], [22]. As these applications operate within a dynamic and heterogeneous environment, the context definition, representation, management and use become important factors that affect and challenge their composition and operation.

In the last few years significant efforts have been devoted to research methods and models for capturing, representing, interpreting and exploiting context information. However, we are still far away from enabling an implicit and intuitive awareness of context, and adaptation to behaviour as efficiently as the human communication practice indicates. Most of the current context-aware systems have been built in an ad-hoc approach, deeply affected by the underlying technology infrastructure utilized to capture the context [7]. To ease the development of context-aware UbiComp applications it is necessary to decouple application composition from context acquisition and representation, and at the same time provide universal models and mechanisms to manage context. The target of this paper is to present an approach for building a context-aware UbiComp system and the context modelling, management and reasoning process that we developed based on this approach.

The structure of the paper is as follows. Section 2 outlines how context is modelled and used in various UbiComp applications focusing on ontology-based context models. In Section 3 we present our approach for building a context-aware UbiComp system. The ontology-based context management and reasoning process that we developed based on this approach is described in Section 4. Lessons learned are presented through a prototype application in Section 5. Finally, Section 6 concludes with a glance on future work.

2. Context-aware UbiComp applications

According to [7] context is: "Any information that can be used to characterise the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves." Context is typically the location, identity and state of people, groups and computation and physical objects, although in UbiComp applications other kinds of context can be used like physical (e.g. location and time), environmental (e.g. weather and light) and personal information (e.g. mood and activity).

A number of informal and formal context models have been proposed in various UbiComp systems; a survey of context models is presented in [16]. Among systems with informal context models, Context Toolkit [6] represents context in form of attribute-value tuples, and Cooltown [12] proposed a Web based model of context in which each object has a corresponding Web description. Both ER and UML models are used for the representation of formal context models in [10]. As ontologies are a promising instrument to specify concepts and their interrelations, they can provide a uniform way for specifying a context model's core concepts as well as an arbitrary amount of subconcepts and facts, altogether enabling contextual knowledge sharing and reuse in a UbiComp system [5]. Thus several research groups have presented ontology-based models of context and used them in UbiComp applications; following we briefly describe the most representative ones.

In the Smart Spaces framework GAIA [14] is presented an infrastructure that supports the gathering of context information from different sensors and the delivery of appropriate context information to UbiComp applications. Their target was to develop a flexible and expressive model for context that can represent the wide variety of possible contexts and support complex reasoning on contexts. So they decided to represent context as first-order predicates written in DAML+OIL. This context model allows deriving new contexts from other sensed context.

The Context Ontology Language (CoOL) [17] is based on the Aspect-Scale-Context Information (ASC) model. Aspects represent classifications (e.g. Temperature), while scales are individual dimensions of aspects (e.g. Celcius). Context information is attached to a particular aspect and scale and quality metadata (e.g. meanError) is associated with information via quality properties. This contextual knowledge is evaluated using ontology reasoners, like F-Logic and OntoBroker. Beyond determination of service interoperability in terms of contextual compatibility and substitutability, this language is used to support context-awareness in distributed service frameworks for various applications.

Wang et al. created an upper ontology, the CONON [18] context ontology, which captures general features of basic contextual entities, a collection of domain specific ontologies and their features in each subdomain. The CONON ontologies are serialized in OWL-DL which has a semantic equivalence to well researched description logics. Thus CONON supports two types of reasoning: reasoning to detect and correct inconsistent context information and reasoning as a means to derive higher level context information. The latter type of reasoning is based on properties like symmetry and transitivity and user-defined rules.

A promising emerging context modelling approach based on ontologies is the COBRA-ONT [2], an ontology for context-aware pervasive computing environments used in the CoBrA system. This system provides a set of OWL ontologies developed for modelling physical locations, devices, temporal concepts, privacy requirements and several other kinds of objects within UbiComp environments. CoBrA employs reasoning for detecting and resolving inconsistent context information, evaluating privacy policies and inferring additional context information based on properties such as temporal and spatial relations. As CoBrA's reasoning over context information is based purely on OWL without additional rule, it is quite limited.

Although each research group follows a different approach for using ontologies in modelling and managing context in UbiComp applications, it has been acknowledged by the majority of researchers [1], [6], [14], [13], [9] that it is a necessity to decouple the process of context acquisition and interpretation from its actual use, by introducing a consistent, reliable and secure context framework which can facilitate the development of context-aware applications.

In this respect, we propose an approach for a context-aware UbiComp system that eases the composition of such applications and separates it from the process of context acquisition. The ontology-based context management and reasoning process that we developed for such a system targets to overcome some limitations of the existing infrastructures presented above. In order to preserve the autonomous nature of artefacts we store an ontology into each artefact and not various ontologies into a server as in GAIA and CoBrA. This solution also overcomes the limitation of systems that demand a specific context ontology for each application. Finally our reasoning process is also enhanced with user-defined rules applied on context stored in each artefact.

3. An approach for building a context-aware UbiComp system

A context-aware application consists of an infrastructure to capture context and a set of rules governing how the application should respond to context changes. In order to isolate the user from the process of context acquisition and management and provide to him a UbiComp system that enables the composition of context-aware applications we propose a system organised in a hierarchy of levels.

Lexical Level: This level examines how signals from the environment, which are captured by sensors, are translated into basic context events or tokens. Generally, any sensor that can contribute to the modelling of the real-world can be considered as source of context. The objective of this level is that context can be modelled in abstraction from sensor technologies and characteristics of particular sensors. The context events can be analysed/combined in the next level in order to represent context phrases or atoms.

Syntactical/Representation Level: The target of this level is to translate context events to meaningful context information (context atoms) by matching sensor data values to real world properties, which have a meaning for some entity of concern (person, place, activity) according to an ontology. In the simplest approach, the representation level can include mappings from data gathered from one or more sensors to context related values defined by an ontology (e.g., the light level is mapped into {low, med, high}).

Reasoning Level: The objective of this level is to provide models for context fusion in context hierarchies. Many sensors and many context atoms are aggregated to infer high-level context properties. For example, the context atoms light, temperature, humidity, noise are combined to infer the high-level context atmosphere {esoteric, exoteric}. Context inference may be seen as a classification problem. In simple cases the reasoning may also be based on the definition of an ontology, which may use simple description logic or user-defined reasoning using first-order logic.

Planning Level: The aim of this level is to define strategies and schedule actions to be taken in response to context variance. We share the same view as described in [8] that usability and predictability of ubiquitous systems arises from a clear and easily grasped relationship between the context structure and the changes in the behaviour we observe. In current systems this relationship is not explicitly articulated but instead exists implicitly in the system's reaction to events. Category theory provides a unified framework to develop semantics of ubiquitous computing that reflects the uncertain and inferential nature of contextual services and their adaptation. The goal is to support the capture of the "behavioural envelope" within which a system can adapt and the strategies for that adaptation.

Interaction Level: The objective of this level is to provide models for personal and collective interactions in AmI environments. A high-level conceptual model separates the low-level context acquisition (lexical level) from the context characterization (representation/reasoning level), from the delivery and reaction to the context (planning level) and from the composition of context-aware UbiComp applications. The developer/user is given a set of abstractions and has to think and design his application in terms of these abstractions. The higher the abstractions the more implementation details are hided.

4. An ontology-based context management and reasoning process

In this section we describe the system that we developed based on the aforementioned approach focusing on its context management and reasoning process.

4.1. A conceptual model for UbiComp applications composition

The key idea behind the proposed conceptual model is that artefacts of AmI environment can be treated as components of a UbiComp application and users can compose UbiComp applications by creating associations between the artefacts. The Plug/Synapse model [21] serves as a common interfacing mechanism between artefacts providing the means to create large scale systems based on simple building blocks. Plugs make visible the artefact's properties, capabilities and services to people and to other artefacts, while synapses are associations between two compatible plugs.

In terms of the application developer, the plugs can be considered as context-providers that offer high-level abstractions for accessing context (e.g., location, state, activity, etc.). For example, an eLamp may have a plug that outputs if the eLamp is switched on or switched off and an eRoom a plug for informing if someone is in this room or not. In terms of the service infrastructure (middleware), they comprise reusable building blocks for context rendering that can be used or ignored depending on the application needs. Each context-provider component reads input sensor data related to the specific application and can output either low level context information such as location, time, light level, temperature, proximity, motion, blood pressure or high-level context information such as activity, environment and mood. An artefact from its own experience and use has two different levels of context; the low level which information acquired from its own sensors and the high level that is an interpretation of its low level context information. Additionally an artefact can get context information from the plugs of other artefacts; this context can be considered as information from a "third-person experience".

The application developers by establishing synapses between plugs both denote their preferences and needs and define the behaviour of the UbiComp application. From the service infrastructure perspective, the synapses determine the context of operation for each artefact; thus each artefact's functionality is adapted to the UbiComp application's structure.

Providing users with this conceptual model, we aim to decouple the low-level context management from the application business logic, which is captured as expressions in terms of high-level concepts that are matched with services available in the current application context. Instead of the classical approach of established interfaces for resource access, this approach decouples the high-level concepts from the instances implemented by each context.

4.2. Context management process

The design approach for composing context-aware UbiComp applications, described in the previous section, needs to be backed by an engineering methodology that defines the correct formulation of the context and behaviour. The proposed context management process is depicted in Fig. 1. The motivation for this process emerged from the fact that artefacts in AmI environment may be in different "states" that change according to the artefacts' use from users and their reaction is based both on users' desires and these states.

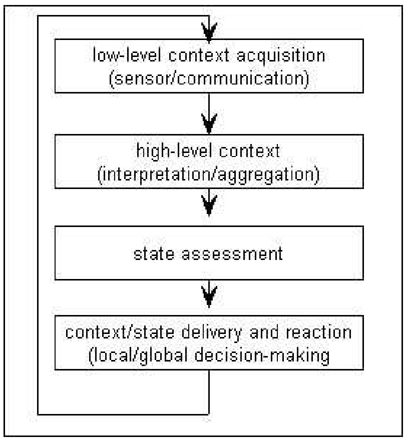
low-level context acquisition
(sensor/communication)
high-level context
(interpretation/aggregation)
state assessment
context/state delivery and reaction
(local/global decision-making

Figure 1: Context management process.

The first step in this context management process is the acquisition of the low level context, which is the raw data given by the sensors (Lexical Level). A set of sensors are attached to an artefact so that to measure various artefact parameters, e.g. the position and the weight of an object placed on an augmented table. As the output of different sensors that measure the same artefact parameter may differ, e.g. sensors may use different metric system, it is necessary to interpret the sensors' output to higher level context information (Syntactical/Representation Level). Aggregation of context is also possible meaning that semantically richer information may be derived based on the fusion of several measurements that come from different homogeneous or heterogeneous sensors. For example, in order to determine if an object is placed on a table requires monitoring the output of table's position and weight sensors.

Having acquired the necessary context we are in a position to assess an artefact's state (Reasoning Level) and decide appropriate response activation (Planning Level). Adopting the definition from Artificial Intelligence, a state is a logical proposition defined over a set of context measurements [15]. This state assessment is based on a set of rules defined by the artefact's creator. The reaction may be as simple as turn on an mp3 player or send an sms to the user or a composite one such as the request of a specific service, e.g. a light service. Such a decision may be based on local context or may require context from external sources as well, e.g., environmental context, location, time, other artefacts. The low (sensor) and high (fused) level data, their interpretation and the local and global decision-making rules are encoded in an ontology.

4.3. An ontology-based context model

The ontology that we created in order to support both the aforementioned conceptual model and the context management process is an enhanced version of the GAS Ontology [3], [4] represented in DAML+OIL. The definition of an artefact's state is emerged from the characterisation of an artefact and it is strongly related to the values of its parameters measured by its sensors. Thus we decided to add the concepts of State and Parameter and define the relations between these concepts and the concept of artefact.

The ontology stored in each artefact is divided into two layers: a common one (Fig. 2) that contains the description of the basic concepts of UbiComp applications and their interrelations and represents the common language among artefacts and a private one that represents artefact's own description as well as its new "knowledge" (experience) has accumulated from its use.

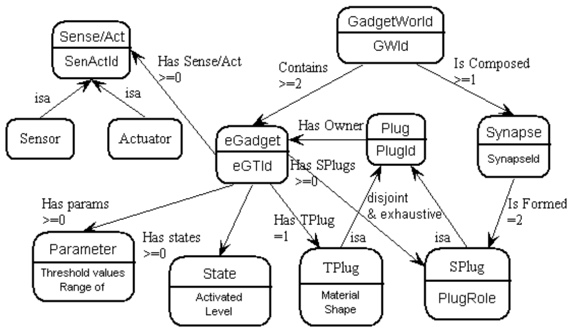

Figure 2: Ontology-based context model.

The basic goal of this ontology is to support a context management process that is based on a set of rules which determine the way that a decision is taken and must be applied on existing knowledge represented by this ontology. The description of the different types of these rules follows.

4.3.1. Rules for artefact's state assessment

The left part of these rules denotes the parameters that affect the state of an artefact and the thresholds or the values for these specific parameters that lead to the activation of the rule, while the right part of these rules denotes the artefact's state that is activated. These rules support the "translation" of low level context (values of parameters measured by sensors) to state assessment; they may also incorporate the translation from low level context to high level context (e.g. perform a set of operations to values measured by sensors like estimate the average value).

4.3.2. Rules for the local decision-making process

These rules exploit exclusively knowledge from the artefact that uses them. Their left part denotes the artefact's states that must be detected and their possible activation level and their right part denotes the artefact's requests and needs. When an artefact has a specific need we can consider that it needs a type of service offered by another artefact. When a rule from this category is activated, the artefact has to search its synapses in order to find a synapse which is associated to another artefact's plug that provides the requested service. If such a synapse is found then the artefact can select it in order to satisfy its need. The situations where more than one synapse is found that may be used to satisfy the request or no synapses are found are handled from the local decision process using another kind of rules. The rules that define the final reaction of an artefact can be defined from the user or can be based on specifically user-defined policies. These rules support both the context delivery and the reaction of an artefact based on the local decision from state assessments.

4.3.3. Rules for the global decision-making process

These rules are similar to the rules for the local decision-making. Their main difference is that the rules for the global decision-making process have to take into account the states of other artefacts and their possible reactions so that to preserve a global state defined by the user.

4.4. Design & Implementation

The architecture of the system that implements the aforementioned context management and reasoning process is on Fig 3. The core modules of this system, Ontology manager, Rule manager and Inference Engine, are part of an updated version of the GAS-OS kernel [11], a middleware that supports the composition of context-aware UbiComp applications and should run on each artefact that participates in such applications.

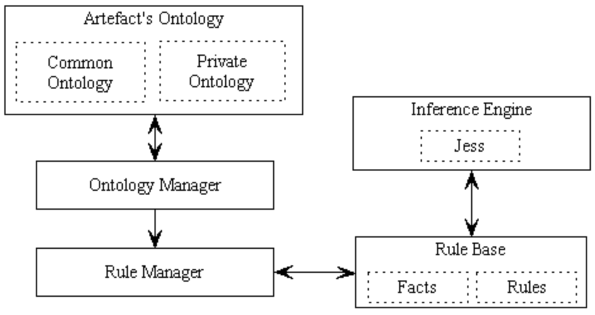

Figure 3: System's architecture.

The Ontology Manager is the responsible module for the manipulation of knowledge represented into the artefact's ontology. Specifically, it can only query the artefact's common ontology, since this ontology cannot be changed during the deployment of an application. On the other hand, it can both query and update the artefact's private ontology. The basic functionality of the Ontology Manager is to provide the other modules of the system with knowledge stored in the artefact's ontology by adding a level of abstraction between them and the ontology.

The Rule Manager is the module that manages the artefact's rule base and it is responsible for both querying and updating this rule base. Note that the rules stored in an artefact's rule base may only contain parameters, states and SPlugs that are defined into the artefact's private ontology. The rule set for a specific artefact can be easily created through a graphical user interface using the Ontology-based Supervisory Logic and Data Acquisition tool. Using this tool, users can easily create rule sets that satisfy the system's constraint that rules should only contain elements defined into the artefact's private ontology. Through this tool users can also change an artefact's rule set dynamically during the participation of the artefact in an application. When the Rule Manager gets an updated rule set from the Ontology-based Supervisory Logic and Data Acquisition tool, it forces the Inference Engine to restart with the updated rule set.

For the initialisation of the context management process apart from the rules a set of initial facts are necessary. The Rule Manager is also responsible for the creation of a file containing the initial facts for an artefact. For example an initial fact may define the existence of an artefact by denoting its parameters, states and SPlugs (reactions) that can participate in its rules and their initial values. In order to create such an initial fact the Rule Manager uses knowledge stored in the artefact's ontology. So it queries the Ontology Manager for any information that it needs, like the artefact's parameters, states and SPlugs.

The Inference Engine supports the decision-making process and it is based on the Jess rule engine (Java Expert System Shell) [23]. In order to initialise its process execution, the Inference Engine needs the artefact's initial facts, which are defined by the Rule Manager, and the rules stored in the rule base. Note that in the current version of our system the rules in the rule base are stored in Jess format. The Inference Engine is informed for all the changes of parameters' values measured by artefact's sensors. When it is informed for such a change it runs all the rules of the rule base. If a rule is activated, this module informs the artefact's operating system for the activation of this rule and for the knowledge that is inferred. The artefact's state and reaction is determined from this inferred knowledge.

5. Lessons learned

In this section we provide an evaluation of our system through the composition of a prototype context-aware UbiComp application and the presentation of the context management process. Our prototype application is an everyday context-aware UbiComp application that aims to keep the user informed about the state of an elderly person. The components of this application are the following:

eHealthyBox: an artefact that monitors the health status of a person based on measurements of sensors attached to person's body. For example, we have used the SelfCheck BP sensor of the Card Guard – The Telemedicine Company; a personal wireless blood pressure and pulse rate monitor. The eHealthyBox based on sensors' measurements and the rules stored into its rule base can decide whether the situation is critical or not. A sample of rules from this rule base is presented in Fig. 4.

```
If (PulseRate > Maria.PR_Threshold) then
EnableEmergentEvent
If (EnableEmergentEvent) then CallAmbulance();
If (EnableEmergentEvent) then NotifyJohn();

If (BodyTemp > 37.5) and (BodyTemp <= 38) then
EnableFeverEvent
If (BodyTemp > 38) and (BodyTemp <= 39) then
EnableFeverAlert
If (BodyTemp > 39) then EnableFeverEmergency
If (EnableFeverEvent) then NotifyMaria();
If (EnableFeverAlert) then NotifyMaria(); NotifyJohn();
If (EnableFeverEmergency) then NotifyMaria();
NotifyJohn(); NotifyDoctor();
```

Figure 4: eHealthyBox rule base.

eMobilePhone: an artefact used for sending sms.

eLamp: a floor lamp used for visual notifications.

eMP3Player: an mp3 player for audio messages.

eDoormat: an augmented rug with pressure sensors attached, used for sensing the occupancy of a house. Based on the sequence the pressure sensors are pressed the eDoormat is capable of deducing if someone is entering or leaving a house, thus if the house is occupied or not.

eMoodCube: an augmented Mathmos lamp with tilt switches attached, used for defining the current status of the user, such as "available", "do not disturb" etc.

According to the scenario Maria, a 78-years old woman, prefers to stay alone in her own house than to live with her son John. Because of the possibility of a heart attack, John wants to be informed about the condition of his mother. The state of Maria's health is defined by a set of sensors attached to her body. The eHealthyBox monitors Maria's state and informs John if her situation is critical and if she needs special care. If Maria's situation is critical the eHealthyBox has to inform John, so it gets from the eDoormat through a synapse the context information if John is in his house or not. If John is in his house, depending on his current status that is determined by the eMoodCube, the eLamp and eMP3Player are used to provide the appropriate visual and acoustic notification respectively to him. If the eMoodCube is set to a "Do not disturb" status, John is notified only for the emergent events. In the case John is out of house, the eMobilePhone will send him an sms informing him about Maria's condition.

Initially John has to compose this application by selecting the artefacts that will take part and setting the synapses between their plugs. For example, John has to set a synapse between the suitable plugs of eHealthyBox and eLamp so that to denote that when his mother's state is critical the eLamp is blinking. For this process John can use an editing tool, like the GWEditor [21], that supports the composition of such applications based on the Plug/Synapse model. The establishment of synapses between two compatible plugs results to the update of both artefacts' private ontologies by the Ontology Manager.

Then John has to set the rules that determine the context management process. This is feasible by using an Ontology-based Supervisory Logic and Data Acquisition tool; a graphical user interface, which provides the user variable operations for viewing the knowledge represented by an artefact's ontology, monitoring an artefact's state and managing an artefact's rules for the context management process. For example John using such a tool may define the rule that denotes that when he is out of house he will be informed about his mother's condition by receiving an sms to his mobile phone. The updates on context management process' rules through this tool conclude to the update of the artefact's rule base by the Rule Manager. When the rule base of an artefact is updated dynamically during its deployment the Rule Manager informs the Inference Engine so that to restart its process with the updated rule base.

The first target of our research was to define an ontology-based context model that decouples the process of context acquisitions and interpretation from its actual use. This target was partially achieved since through the Plug/Synapse model the user has just to denote the sources of context that artefacts can exploit and to define the interpretation of this context through rules. This may become difficult especially for defining the rules of global decision making. So we believe that users need a tool that permits the definition of abstract rules for the whole application and creates the necessary ones to be stored in each artefact of the application.

The second target of our work was to implement the proposed context management and reasoning process. The module that is responsible for this reasoning process is the Inference Engine. Assume the rules of eHealthyBox presented in Fig. 4. Any change to the low level context of eHealthyBox (raw data from its sensors) informs the Inference Engine and generates a new "run". If the value of the pulse rate is above the threshold then the first three rules will be activated. Then the Inference Engine will inform the operating system of the eHealthyBox for the inferred knowledge; specifically it will define the message that must be sent through an SPlug. The other artefacts of our prototype will be informed for the context emerged from the state of eHealthyBox, which represents Maria's condition, through the synapses between their SPlugs and the eHealthyBox's SPlug. From experiments that we worked out we concluded that the inference mechanism that we developed and defines an artefact's state and reaction based on context information is fairly efficient for a number of forty rules.

Using our context management and reasoning process such applications can be dynamically changed either by adding or removing artefacts, as each artefact acquires and manages context separately based on its rule set. Finally the reasoning process permits user-defined rules that can be dynamically updated.

6. Future Work

In this paper we presented an ontology-based context modelling, management and reasoning process that was developed for the composition of context-aware UbiComp applications from AmI artefacts.

Our next steps for the future are to represent the artefacts' ontologies in OWL (Web Ontology Language) and the rules in SWRL (Semantic Web Rule Language) and try use the Jena framework for inference and reasoning. Additionally, we want to introduce a degree of uncertainty in the reasoning process based on the reliability of sensors' measurements and by adding confidence values to the rules' activation part. Finally, end-users may not need to configure each artefact separately (e.g. setting its rules), but to view and manage the whole application in an abstract way. The system and the context management process that we have developed can support both these approaches, but enhanced end-users tools focusing on context handling are necessary.

References

[1] Biegel, G., Cahill, V. (2004) *A Framework for Developing Mobile, Context Aware Applications. in 2nd IEEE Conference on Pervasive Computing and Communications*. Orlando, FL, March 14-17.

[2] Chen, H., Finin, T., Joshi, A. (2004) *An ontology for context aware pervasive computing environments.* Knowledge Engineering Review - Special Issue on Ontologies for Distributed Systems, Cambridge University Press.

[3] Christopoulou, E., Kameas, A. (2005) *GAS Ontology: an ontology for collaboration among ubiquitous computing devices. in International Journal of Human – Computer Studie: special issue on Protégé*. Vol. 62, issue 5.

[4] Christopoulou, E., Kameas, A. (2004) *Using ontologies to address key issues in ubiquitous computing systems. in 2nd European Symposium on Ambient Intelligence*. Eindhoven, the Netherlands. LNCS, Vol. 3296, pp 13-24.

[5] De Bruijn, J. (2003) *Using Ontologies – Enabling Knowledge Sharing and Reuse on the Semantic Web. Technical Report DERI-2003-10-29*. Digital Enterprise Research Institute (DERI), Austria.

[6] Dey, A. K., Salber, D., Abowd, G. D. (2001) *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications*, *Human-Computer Interaction Journal*, Volume 16 (2-4), pp. 97-166.

[7] Dey, A. K. (2001) *Understanding and using context, Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing* 5, 1.

[8] Dobson, S., Nixon, P. (2004) *More principled design of pervasive computing systems, in Proceedings of Engineering for Human-Computer Interaction and Design, Specification and Verification of Interactive Systems*. Hamburg, Germany.

[9] Henricksen, K., Livingstone, S., Indulska, J. (2004) *Towards a Hybrid Approach to Context Modelling, Reasoning and Interoperation. in the proc. of the 1st International Workshop on Advanced Context Modeling, Reasoning and Management, 6th UBICOMP*. Nottingham, UK. pp. 54-61.

[10] Henricksen, K., Indulska, J., Rakotonirainy, A. (2002) *Modeling Context Information in Pervasive Computing Systems, In F. Mattern and M. Naghshineh, editors, Pervasive 2002*, pp. 167– 180, Springer Verlag, Berlin.

[11] Kameas, A., et al. (2003) *An Architecture that Treats Everyday Objects as Communicating Tangible Components. in 1st IEEE International Conference on Pervasive Computing and Communications*. Fort Worth, USA.

[12] Kindberg T. et al. (2000) *People, Places, Things: Web Presence for The Real World, Technical Report HPL-2000-16*, HP Labs.

[13] Nixon, P. et al. (2002) *Engineering context-aware enterprise systems. in Workshop on Engineering Context-Aware Object-Oriented Systems and Environments.* Seattle, USA.

[14] Ranganathan, A., Campbell, R. (2003) *An infrastructure for context-awareness based on first order logic. Personal and Ubiquitous Computing*. 7(6):353–364.

[15] Russell, S., Norvig, P. (2003) *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition.

[16] Strang, T., Linnhoff-Popien, L. (2004) A Context Modeling Survey. in 1st International Workshop on Advanced Context Modelling, Reasoning And Management, Nottingham, *6th International Conference on Ubiquitous Computing*. UK. pp. 33-40.

[17] Strang, T., Linnhoff-Popien, L., Frank, K. (2003) *CoOL: A Context Ontology Language to enable Contextual Interoperability. in LNCS 2893: proc. of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems.* Paris, France. pp. 236–247.

[18] Wang, X. H. et al. (2004) *Ontology Based Context Modeling and Reasoning using OWL, Workshop on Context Modeling and Reasoning* at IEEE International Conference on Pervasive Computing and Communication. Orlando, Florida.

[19] Weiser, M. (1991) *The Computer for the 21st Century. Scientific American*. 265, pp. 94-10.

[20] Disappearing Computer initiative website http://www.disappearing-computer.net/

[21] extrovert-Gadgets project website http://www.extrovert-gadgets.net

[22] IST Advisory Group (ISTAG) (2001): *Scenarios for Ambient Intelligence in 2010.* http://www.cordis.lu/ist/istag-reports.htm

[23] Jess - the Rule Engine for the Java Platform http://herzberg.ca.sandia.gov/jess/