Using Ontologies to Address Key Issues in Ubiquitous Computing Systems

Eleni Christopoulou and Achilles Kameas

Research Academic Computer Technology Institute, Research Unit 3, Design of Ambient Intelligent Systems Group, 61 Riga Feraiou str. 26221, Patras, Greece
{hristope, kameas}@cti.gr

Abstract. One proposed way to realize the AmI vision is to turn everyday objects into artifacts (by adding sensing, computation and communication abilities) and then use them as components of Ubiquitous Computing (UbiComp) applications within an Ami environment. The (re)configuration of associations among these artifacts will enable people to set up their living spaces in a way that will serve them best minimizing at the same time the required human intervention. During the development and deployment of UbiComp applications, a number of key issues arise such as semantic interoperability and service discovery. The target of this paper is to show how ontologies can be used into UbiComp systems so that to address such issues. We support our approach by presenting the ontology that we developed and integrated into a framework that supports the composition of UbiComp applications.

1 Introduction

One proposed way to realize the AmI vision is to turn everyday objects into artifacts (by adding sensing, computation and communication abilities) and then use them as components of UbiComp applications within AmI environments. The (re)configuration of associations among these artifacts will enable people to set up their living spaces in a way that will serve them best minimizing at the same time the required human intervention. A limitation of the current technology is that it requires human intervention, in order to enable the communication and collaboration among the artifacts.

Within the context of an architectural framework that supports the composition of UbiComp systems the heterogeneity of the devices that constitute the artifact "ecologies" is an important parameter. So the feasibility of semantic interoperability among heterogeneous devices is a key issue that arises. Also, the dynamic nature of UbiComp applications that may lead to unanticipated situations requires the existence of a service discovery mechanism. Various types of middleware (based on CORBA, Java RMI, SOAP, etc.) have been developed so that to enable the communication between different UbiComp devices. However, these middleware have no facilities to handle issues like the semantic interoperability among heterogeneous artifacts.

P. Markopoulos et al. (Eds.): EUSAI 2004, LNCS 3295, pp. 13–24, 2004.

In order to handle such issues the primary requirement is to provide to the heterogeneous devices a common language that supports the communication and collaboration among them. This common language must be based on the description and definition of the basic concepts of the artifact "ecologies" and must be both flexible and extensible so that new concepts can be added and represented. The target of this paper is to present the developed ontology that represents this common language and show how this ontology can handle the key issues that arise during the development and deployment of UbiComp applications.

The rest of the paper is organised as follows. Section 2 describes the basic concepts of the framework that supports the composition of UbiComp applications, the key issues that arise during this procedure and how an ontology can accommodate them. Section 3 presents the ontology that was designed and developed and section 4 introduces the mechanism that was developed for the management of this ontology. Section 5 describes the use of this ontology into UbiComp applications through examples based on a specific scenario. In section 6 related approaches for ontologies in ubiquitous computing environments are presented. The paper closes with the lessons learned from our experience, an evaluation of our approach and an outlook on future work in section 7.

2 Key Issues in Ubiquitous Computing Systems

The Gadgetware Architectural Style (GAS) [5] is an architectural framework that supports the composition of UbiComp applications from everyday physical objects enhanced with sensing, acting, processing and communication abilities. UbiComp applications are dynamic, distinguishable, functional configurations of associated artifacts, which communicate and/or collaborate in order to realize a collective behavior. Each artifact makes visible its properties, capabilities and services through specific interfaces (we'll sometimes use the term "Plugs"); an association between two compatible interfaces is called a "Synapse".

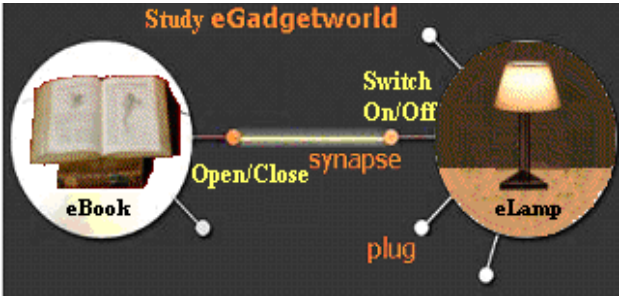

Fig. 1. A study UbiComp application realized as a synapse between plugs

The basic concepts defined above are illustrated in Figure 1 through a simplified scenario. Two artifacts (eBook and eLamp) are connected through a synapse which is established between two plugs forming a "study" UbiComp application. When the

user opens the eBook, the eLamp switches on, adjusting the light conditions to a specified luminosity level in order to satisfy the user's profile. Each artifact operates independently of the other. Two plugs are being used in this scenario: "open/close" reflecting the state of the book and "switch on/off" reflecting the state of the lamp. Although not obvious, these plugs are compatible. Thus, an end-user can compose a simple UbiComp application by establishing a synapse between these two plugs. Different users may have access to this "application" each one having defined his own study profile. Even during the composition of such a simple UbiComp application a number of key issues that must be addressed arise. Following we present a set of such issues and explain our decision to use ontologies in order to address them.

2.1 Semantic Interoperability Among Artifacts

The composition of UbiComp applications is based on the interaction among devices. Since the heterogeneity of these devices is an aspect that cannot be neglected, the challenge that we have to handle is the feasibility of semantic interoperability among autonomous and heterogeneous devices. In our approach and in order to present the autonomous nature and function of each artifact, we chose to base this interaction on well-defined and commonly understood concepts, so that the artifacts can communicate with each other in a consistent and unambiguous way. So the artifacts have to use the same language and a common vocabulary. Note that this common language must be flexible and extensible so that new concepts can be added and represented. For the representation of this common language we decided to use an ontology that describes the semantics of the basic terms of UbiComp applications and defines their inter-relations.

2.2 Dynamic Nature of UbiComp Applications

One of the most important features of UbiComp applications is that they are created in a dynamic way. Users are permitted to create and delete synapses between artifacts whenever they want without restrictions. As synapses are associations between two compatible plugs, the creation of a synapse requires some form of plugs compatibility check. The compatibility of two plugs is determined by several factors e.g. the type of input that they accept and the type of output that they produce, that must be represented into a formal form.

The dynamic nature of UbiComp applications depends also on artifacts mobility that can cause the dynamic disestablishment of a synapse. For example the disestablishment of a synapse may happen when two artifacts move outside of each other's range or when an artifact suddenly "disappears" due to low battery or other failure. Since our vision refers to "smart" UbiComp applications and artifacts that exploit the knowledge that they have acquired by experience, the desirable solution to the "disappearance" of an artifact is to automatically replace it and not to just ignore it. In order to ensure artifacts replacement feasibility a mechanism for finding "similar" artifacts should be described. We selected to replace an artifact with another one that offers the same services.

2.3 Semantic Service Discovery

The plugs are software classes that make visible the artifacts' capabilities to people and to other artifacts. The term that we use for these capabilities is Services. For example the artifact "eLamp", Figure 1, through the Plug "switch on/off" provides the Service "light". The concept of services in UbiComp applications is a fundamental one, since services can play a major role when determining artifacts' replaceability and plugs' compatibility; for example we assume that an artifact A participating in Synapse S with Plug P can be replaced by another artifact B that provides a service P' similar to P. Furthermore, a user forms synapses seeking to achieve certain service configurations; thus a service discovery mechanism is necessary. In UbiComp environments this mechanism must be enhanced to provide a semantic service discovery; with this term we refer to the possibility to discover all the semantically similar services. This mechanism is assisted by a service classification represented into the ontology that we developed.

2.4 Conceptualisation of Ubicomp Applications

The GAS constitutes a generic framework shared by both artifact designers and users for consistently describing, using and reasoning about a family of related UbiComp applications. GAS defines the concepts and mechanisms that will allow people to define, create UbiComp applications out of artifacts and use them in a consistent and intuitive way. As the artifacts are enhanced everyday physical objects, users do not have to deal with unfamiliar to them objects, but merely to view their world from a different perspective and get familiar with its enhanced concepts. This new world view is constituted of a set of basic terms, their definitions and their inter-relations. The necessity of capturing and representing this knowledge is evident, as the deployment of UbiComp applications is based on this knowledge. Since ontologies can conceptualise a world view by capturing the general knowledge and defining the basic concepts and their interrelations [15], we decided to use an ontology in order to conceptualise the terms of UbiComp applications. The ontology that we developed is the GAS Ontology and its first goal was the description of the semantics of the basic terms of the UbiComp applications, such as eGadget (our term for artifact), Plug, Synapse, Service, eGadgetWorld (our term for UbiComp application), and the definition of their interrelations.

2.5 Context-Awareness

An important issue of UbiComp environments is the context-awareness, as these environments must be able to obtain the current context and adapt their behavior to different situations. In UbiComp applications, different kinds of context can be used like physical information, e.g. location and time, environmental information, e.g. weather and light, personal information, e.g. mood and activity. In our case, the term context refers to the physical properties of artifacts including their sensors/actuators and to their plugs that provide services; for example the eBook artifact through its

plug "open/close" provides to other artifacts a kind of context information relative to its state. The user, by establishing synapses between plugs, defines the emerging behavior of a UbiComp application; e.g. the user with the synapse at the "study" UbiComp application, illustrated in Figure 1, defines the eLamp's behavior in proportion to the context provided by the eBook. Thus the UbiComp applications can demonstrate different behaviors even with the same context information.

3 Designing the GAS Ontology

The ontology that we developed in order to address the aforementioned issues in ubiquitous computing systems is the GAS Ontology [2] and is written in DAML+OIL. The basic goal of the GAS Ontology is to provide the necessary common language for the communication and/or collaboration among the artifacts.

3.1 Ontology Layers

The artifacts' ontology contains the description of the basic concepts of UbiComp applications and their inter-relations; for the feasible communication among artifacts this knowledge must be common. Additionally an artifact's ontology must both contain artifact's description; e.g. the description of its plugs and services, and represent its acquired knowledge emerged from the synapses that its plugs participate to. So the knowledge that each artifact's ontology represent cannot be the same for all artifacts, as it depends on the artifact's description and on the UbiComp applications that the artifacts has participated in the past.

Since artifact's interoperability is based on their ontologies, the existence of different ontologies could result to inefficient interoperability. An awkward solution to this issue could be the merging of all existing artifacts' ontologies into a global one that would inevitably result into a very large knowledge base. This solution is undesirable for two reasons; first it does not respect the limited memory capabilities of the artifacts and second it would work properly if all artifacts ontologies were synchronized. Another solution could be the use of a server into which all artifacts' ontologies are stored and each artifact can have access to it. This solution conflicts with the autonomous nature of artifacts.

The solution that we propose allows each artifact to have a different ontology with the condition that all ontologies will be based on a common vocabulary. Specifically the GAS Ontology is divided into two layers: the GAS Core Ontology (GAS-CO); that contains the common vocabulary, and the GAS Higher Ontology (GAS-HO); that represents artifact's specific knowledge using concept represented into GAS-CO.

3.2 The GAS Core Ontology (GAS-CO)

The GAS-CO describes the common language that artifacts use to communicate. So it must describe the semantics of the basic terms of UbiComp applications and define their inter-relations. It must also contain the service classification in order to support the service discovery mechanism. An important feature of the GAS-CO is that it

contains only the necessary information for the interoperability of artifacts in order to be very small and even artifacts with limited memory capacity may store it. The GAS-CO is static and it cannot be changed either from the manufacturer of an artifact or from a user. The graphical representation of the GAS-CO is on Figure 2.

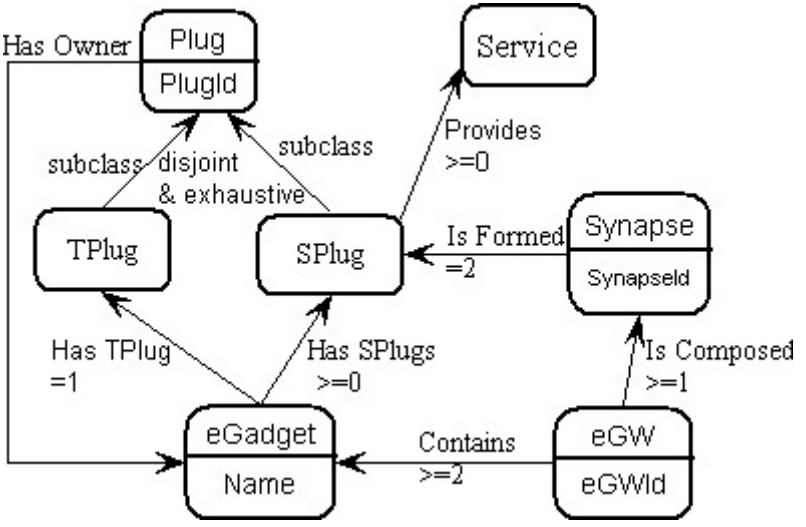

Fig. 2. A graphical representation of GAS-CO

The core term of GAS is the eGadget (eGt). In GAS-CO the eGt is represented as a class, which has a number of properties, like name etc. The notion of plug is represented in the GAS-CO as another class, which is divided into two disjoint subclasses; the TPlug and the SPlug. The TPlug describes the physical properties of the object that is used as an artifact like its shape; note that there is a cardinality restriction that an artifact must have exactly one TPlug. On the other hand an SPlug represents the artifact capabilities and services; artifacts have an arbitrary number of SPlugs. Another GAS-CO class is the synapse that represents a synapse among two plugs; a synapse may only appear among two SPlugs. Using the class of eGW the GAS-CO can describe the UbiComp applications that are created by the users; an eGW is represented by the artifacts that contains and the synapses that compose it. The class of eGW has two cardinality constraints; an eGW must contain at least two artifacts and a synapse must exist between their SPlugs.

As an eGt through an SPlug provides a number of services, the GAS-CO contains a class for the notion of service. An artifact's services are close related to what the artifact's actuators/sensors can transmit/perceive. So the class of service is divided into subclasses so that to describe a service classification, which is based on the type of the signals that an actuator/sensor transmits/perceives. Some elementary forms of signals that are described are the following: electric, electromagnetic, gravity, kinetic, optic, thermal and sonic. Note that a service can be further refined into higher level services: e.g. the optic service can be refined into light, image, etc. Additionally a service may have a set of properties; e.g. light can have as properties the color, the luminosity, etc.

3.3 The GAS Higher Ontology (GAS-HO)

The GAS-HO represents both the description of an artifact and its acquired knowledge. These descriptions follow the definitions contained in the GAS-CO. So, specifically the knowledge stored into the GAS-HO is represented as instances of the classes defined into the GAS-CO. For example the GAS-CO contains the definition of the concept SPlug, while the GAS-HO contains the description of a specific SPlug represented as an instance of the concept SPlug. Note that the GAS-HO is not a stand-alone ontology, as it does not contain the definition of its concepts and their relations.

Since the GAS-HO represents the private knowledge of each artifact, it is different for each artifact. Therefore we can envision GAS-HO as artifact's private ontology. Contrary to GAS-CO, which size is required to be small enough, the size of GAS-HO can depend only on artifact's memory capacity. Obviously GAS-HO is not static and it can be changed over time without causing problems to artifacts communication. As the GAS-HO contains both static information about the artifact and dynamic information emerged from its knowledge and use, we decided to divide it into the GAS-HO-static and the GAS-HO-volatile.

The GAS-HO-static represents the description of an artifact containing information about artifact's plugs, the services that are provided through these plugs, its sensors and actuators, as well as its physical characteristics. For example, the GAS-HO-static of the "eLamp" artifact contains the knowledge about the physical properties of "eLamp", such as its luminosity, the description of its SPlug "switch on/off" based on the definition provided by GAS-CO, as well as the declaration that the SPlug "switch on/off" provides the service "light".

On the other hand the GAS-HO-volatile contains information derived from the artifact's acquired knowledge and its use. Specifically it describes the synapses which the artifact's plugs are connected to, the UbiComp applications which it takes part to, as well as information about the capabilities of other artifacts that has acquainted through communication. An artifact's GAS-HO-volatile is updated during the artifact's various activities, like the establishment of a new synapse.

4 The GAS Ontology Manager

An artifact in order to participate in our UbiComp applications has to be GAS-compatible. An artifact is GAS-compatible if it uses the GAS-Operating System (GAS-OS), which is responsible for the communication among artifacts through a communication module and the management of synapses through the process manager. The GAS Ontology manager is a module of the GAS-OS and provides the mechanism for the interaction of an artifact with its stored ontology and the management of this ontology.

One of the most important features of the GAS Ontology manager is that it adds a level of abstraction between GAS-OS and the GAS Ontology. This means that only the GAS Ontology manager can understand and manipulate the GAS Ontology; the GAS-OS can simply query this module for information stored into the GAS Ontology without having any knowledge about the ontology language or its structure. Therefore

any changes to the GAS Ontology affect only the GAS Ontology Manager and the rest of the GAS-OS is isolated from them.

The GAS-CO must be common for all the artifacts and it cannot be changed during the deployment of UbiComp applications. So the GAS Ontology manager provides methods that can only query the GAS-CO for knowledge such as the definitions of specific concepts like eGadget and Plug and knowledge relevant to the service classification. Likewise it can only query the GAS-HO-static of an artifact. On the other hand since it is responsible for keeping up to date the GAS-HO-volatile of an artifact, it can both read and write to it. As the GAS-HO contains only instances of the concepts defined in the GAS-CO, the basic methods of the GAS Ontology manager relevant to the GAS-HO can query for an instance and add new ones based on the concepts defined in the GAS-CO. So an important feature of the GAS Ontology manager is that it enforces the integrity of the instances stored in the GAS-HO with respect to the concepts described in GAS-CO.

The communication among artifacts is initially established using the artifacts' GAS-HO; if their differences obscure the communication, the GAS Ontology manager is responsible for the interpretation of GAS-HO based on the common GAS-CO. Therefore the communication among artifacts is ensured. Apart from assisting the communication among artifacts, the GAS Ontology manager enables knowledge exchange among them by sending parts of an artifact's GAS-HO to another's.

One of the GAS Ontology goals is to describe the services that the artifacts provide and assist the service discovery mechanism. In order to support this functionality, the GAS Ontology manager provides methods that query both the GAS-HO-static and the GAS-HO-volatile for the services that an SPlug provides as well as for the SPlug that provide a specific service. Thus the GAS Ontology manager provides to the GAS-OS the necessary knowledge stored in an artifact's ontology relevant to the artifact's services, so that to support the service discovery mechanism. Similarly the GAS Ontology manager can answer queries for plugs compatibility and artifacts replaceability.

5 Using the GAS Ontology in a Ubiquitous Computing System

In this section we present an example of how we can use the GAS Ontology into a ubiquitous computing environment and the role of the GAS Ontology manager, using the scenario for the study UbiComp application illustrated in Figure 1. According to this scenario a user creates its own "study" UbiComp application using two artifacts, an eBook and an eLamp.

In the UbiComp applications the interaction among artifacts is feasible because it is based on common concepts and terms. These terms are defined into the GAS-CO. So as the GAS-CO provides the artifacts with the necessary common language, both the eBook and the eLamp artifacts have stored the same GAS-CO.

On the other hand the artifacts' GAS-HO ontologies are different. For example the eLamp's GAS-HO-static contains information about eLamp's SPlug "switch on/off" and the eBook's GAS-HO-static contains the description of SPlug "open/close". These two artifacts are connected through a synapse which is established between the

aforementioned plugs forming the study UbiComp application. So when the user opens the eBook, the eLamp switches on, adjusting the light conditions to a specified luminosity level in order to satisfy the user's profile. The knowledge emerged from this synapse is stored in the GAS-HO-volatile of both the artifacts that participate to it. So the eBook "knows" that it's SPlug "open/close" participates to a synapse with an SPlug that provides the service "light" with a specific luminosity. Note that the GAS Ontology manager is responsible for storing knowledge into an artifact's GAS-HO-volatile by using the definitions of the concepts represented in the GAS-CO.

As the context information that is used in the UbiComp applications describes the physical and digital properties of artifacts, it is represented into both the GAS-CO and each artifact's GAS-HO-static. Note that the developer of the ubiquitous computing system has access to this information and can change it or add new elements. The GAS-HO-volatile of artifacts contains mainly knowledge emerged from the synapses that compose an UbiComp application. So this information represents the artifacts' behavior when they get context information through their synapses; these behaviors are defined by the user of an UbiComp application. As the GAS Ontology contains both context information and the description of the behaviors in proportion to context, makes the UbiComp applications context-aware environments.

If this synapse is broken, for example because of a failure at the eLamp, a new artifact having an SPlug that provides the service "light" must be found. The eBook's GAS-OS needs to find another artifact with an SPlug that provides the service "light". The eBook's GAS-OS is responsible to send a message for service discovery to the other artifacts' GAS-OS that participate to the same UbiComp application. This type of message is predefined and contains the type of the requested service and the service's attributes. Note that an artifact may query just for type of service or for a service with specific attributes. Above we showed that an artifact can be replaced by another one providing similar services. As the GAS Ontology contains both physical and digital information for an artifact it is easy to exploit such information in order to replace an artifact.

When the GAS-OS of an artifact receives a service discovery message, it forwards it to the artifact's GAS Ontology manager. Assume that the artifact "eDeskLamp" participates to the "study" UbiComp application and that this is the first artifact that gets the message for service discovery. The eDeskLamp's GAS Ontology Manager first queries GAS-HO-static of eDeskLamp in order to find if this artifact has an SPlug that provides the service "light". If we assume that the eDeskLamp has the SPlug "LampDimmer" that provides the service light, the GAS Ontology manager will send to the eDeskLamp's GAS-OS a message with the description of this SPlug. If such an SPlug is not provided by the eDeskLamp, the GAS Ontology Manager queries the eDeskLamp's GAS-HO-volatile in order to find if another artifact, with which the eDeskLamp has previously collaborated, provides such an SPlug. If the GAS Ontology Manager finds into GAS-HO-volatile such an SPlug it sends to the artifact's GAS-OS the description of this SPlug. If the queried artifact, in our example the eDeskLamp, has no information about an SPlug that provides the requested service, the control is sent back to GAS-OS, which is responsible to send the query message for the service discovery to another eGadget. Note that all artifacts have the same service classification, which is stored into the GAS-CO; thus the messages for service discovery are based on this classification.

6 Related Work

Since the artifacts can be perceived as agents that communicate and collaborate, our work is closely related to the field of agent communities. It is widely acknowledged that without some shared or common knowledge the members of a multi-agent system have little hope of effective communication. The solution that we propose is based on the idea that all artifacts have a common ontology, the GAS-CO, and each artifact have also a different, "private" ontology, the GAS-HO that is based on the GAS-CO. This idea is similar to the one presented in [14], where the communication among the agents relies on partially shared ontologies.

Ontologies have been used in a number of ubiquitous computing infrastructures in order to address issues emerged from the composition of ubiquitous computing systems. A known use case is the UbiDev [5] a homogeneous middleware that allows definition and coordination of services in interactive environment scenarios. In this middleware, according to [12], resource classification relies on a set of abstract concepts collected in an ontology and the meaning of these concepts is implicitly given by classifiers. The main advantage of this approach in facing resource management problem is that resources selection is based on their semantics that is given by the context. Since every application may have its ontology the application structure results separated from the implementation. This approach is different than the one that we have followed, because whereas they use an ontology for each application that includes several devices, our goal is to provide an ontology that facilitates the use of devices in various ad hoc UbiComp applications.

Ontologies are also integrated in the Smart Spaces framework GAIA [8] [11]. In this work the ontologies have been used in order to overcome a number of problems in the GAIA Ubiquitous computing framework, such as the interoperability between different entities, the discovery and matching and the context-awareness. The approach that the GAIA framework follows is fairly different to the one that we have proposed for the eGadgets project. Specifically in the GAIA framework there is an Ontology Server that maintains the ontologies and there are different kinds of ontologies, such as ontologies that have meta-data about the environment's entities and ontologies that describe the environment's contextual information. The ontologies in GAIA are also used in order to support the deployment of context-aware ubiquitous environments [10]. Another approach is the COBRA-ONT [1], an ontology for context-aware pervasive computing environments.

The Task Computing Environment [7] was implemented in order to support the task computing that fills the gap between what users really want to do and the capabilities of devices and/or services that might be available in their environments. This approach is fairly different to ours, since they use the OWL-S so that to describe the Web services and the services offered by the devices.

Finally a very interesting work is the one made by the Semantic Web in UbiComp Special Interest Group [13]. The basic goal of this group is to define an ontology to support knowledge representation and communication interoperability in building pervasive computing applications. This project's goal is not aimed to construct a comprehensive ontology library that would provide vocabularies for all possible pervasive applications, but to construct a set of generic ontologies that allow developers to define vocabularies for their individual applications.

7 Lessons Learned and Future Work

In this paper we outlined a set of key issues that arise during the composition of UbiComp applications and presented how ontologies can be used so that to address such issues. We supported our approach by describing the GAS Ontology that we developed and integrated into the GAS, a framework that supports the deployment of UbiComp applications using the artifacts as building blocks. Although in this paper we showed through simple examples the use of the GAS Ontology in UbiComp applications, this ontology has been used so that to compose and deploy a number of UbiComp applications. Till now we have used this infrastructure in order to build UbiComp applications with more that ten artifacts and approximately nine synapses between their plugs.

Additionally we used the GAS Ontology in various demonstrations where non-experienced users created their own UbiComp applications, so that to evaluate it in demanding situations. For example, during demonstrations users tried to establish synapses between incompatible plugs. Such situations were successfully handled from the GAS Ontology manager by using the knowledge represented into the ontology so that to check the plugs' compatibility. As these demonstrations went on for many hours the disestablishment of synapses due to artifacts' failure and mobility was a frequent event. The infrastructure's reaction in these events was the discovery of artifacts that provide semantically similar services. Although the service discovery mechanism always proposed an appropriate artifact, the current version of the GAS Ontology is restrictive since it demands all artifacts to have the same service classification. This is a limitation that we intent to eliminate by adding to GAS Ontology manager the capability to map a service description to another one, using the knowledge that artifacts have acquired from their collaboration.

The design of the GAS Ontology and the approach to divide it into two layers resulted to be very helpful for both its development and use. Specifically this approach resulted to a small sized GAS-CO allowing the creation of large, extensible and flexible GAS-HOs. So it satisfies the demands of long-running and real-time UbiComp systems. During the construction of artifacts' GAS-HO the difficulty that we encountered was relevant to the definition of the services that plugs provide. For example the eBook's plug "open/close" reflects its state but it can also be regarded as a plug that provides the service "switch". In order to ease the creation of GAS-HOs, one of our goals is to create a graphical interface through which users also can add information emerged from their own perception and demands.

Regarding the issue of context-awareness our infrastructure is on an early stage. We believe that the use of plug/synapse model as a context model is sufficient, although we need a more elaborate context management and reasoning process. The first step is the acquisition of the low-level context; raw data from sensors, and then their interpretation to high-level context information. Then artifacts based on their context will assess their state and select their appropriate behaviour using a set of rules and axioms. The reasoning will be based on the definition of the ontology, which may use simple description logic or first-order logic. Finally one of our goals is to define a user model so that to handle the existence of various users' profiles into the same UbiComp application.

Acknowledgements. The research described in this paper has been carried out in "extrovert-Gadgets" [4], a research project funded in the context of EU IST/FET proactive initiative "Disappearing Computer" – IST-2000-25240. This work is carrying on in the the EU IST/FET funded project PLANTS [9]. The authors wish to thank fellow researchers from both projects' partners.

References

1. Chen, H., Finin, T., Joshi, A., 2004. An ontology for context aware pervasive computing environments, To appear, Knowledge Engineering Review - Special Issue on Ontologies for Distributed Systems, Cambridge University Press.
2. Christopoulou, E., Kameas, A., 2004. GAS Ontology: an ontology for collaboration among ubiquitous computing devices, to appear in Protégé special issue of the International Journal of Human – Computer Studies.
3. Disappearing Computer initiative http://www.disappearing-computer.net/
4. extrovert-Gadgets project website http://www.extrovert-gadgets.net
5. Kameas, A., Bellis, S., Mavrommati, I., Delaney, K., Colley, M., Pounds-Cornish, A., 2003. An Architecture that Treats Everyday Objects as Communicating Tangible Components. In Proceedings of the 1st IEEE International conference on Pervasive Computing and Communications (PerCom03). Forth Worth, USA
6. Maffioletti, S., Hirsbrunner, B., 2002. UbiDev: A Homogeneous Environment for Ubiquitous Interactive Devices. Short paper in Pervasive 2002 - International Conference on Pervasive Computing. pp. 28-40. Zurich, Switzerland.
7. Masuoka, R., Labrou, Y., Parsia, B., Sirin, E., Ontology-Enables Pervasive Computing Applications, IEEE Intelligent Systems, Sept/Oct 2003, pp 68-72.
8. McGrath, R., Ranganathan, A., Campbell, R., Mickunas, D., 2003. Use of Ontologies in Pervasive Computing Environments. Report number: UIUCDCS-R-2003-2332 UILU-ENG-2003-1719 Department of Computer Science, University of Illinois
9. PLANTS project website http://plants.edenproject.com
10. Ranganathan, A., Campbell, R., 2003. A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. In ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil.
11. Ranganathan, A., McGrath, R., Campbell, R., Mickunas, D., 2003. Ontologies in a Pervasive Computing Environment. Workshop on Ontologies in Distributed Systems at IJCAI, Acapulco, Mexico.
12. Schubiger, S., Maffioletti, S., Tafat-Bouzid, A., Hirsbrunner, B., 2000. Providing Service in a Changing Ubiquitous Computing Environment. Proceedings of the Workshop on Infrastructure for Smart Devices - How to Make Ubiquity an Actuality, HUC.
13. Semantic Web in UbiComp Special Interest Group. http://pervasive.semanticweb.org
14. Stuckenschmidt, H., Timm, I. J., 2002. Adapting Communication Vocabularies using Shared Ontologies. In the Proceedings of the Second International Workshop on Ontologies in Agent Systems (OAS). Bologna, Italy.
15. Uschold, M., Gruninger, M., 1996. Ontologies: principles, methods and applications. Knowledge Engineering Review, Vol. 11 No. 2, pp. 93-155.