A Prototype of a Context-Based Architecture for Intelligent Home Environments

Pablo A. Haya, Germán Montoro, and Xavier Alamán

Universidad Autónoma de Madrid
Dpto. Ingeniería Informática
Ctra. Colmenar Viejo km 15.
Madrid 28049 – SPAIN
{Pablo.Haya, German.Montoro, Xavier.Alaman}@uam.es

Abstract. This paper presents a proposal of a context-based architecture to achieve the required synergy among the ubiquitous computing devices of an intelligent environment. These devices produce context information that models the behaviour of the environment. This context information is the glue among the devices and the context-aware applications. The generated context information provides a common view of the world. A blackboard architecture allows to share this context information and a context model is proposed to represent it. A prototype of such a smart room has been developed, including several devices as well as a set of context-aware demonstrators. They work together employing the context information stored on the blackboard.

1 Introduction

In the last years, numerous research groups have been working in different technologies related with what Weiser defined as "Ubiquitous Computing" [1]. Weiser's vision[1] stands on three key points: Firstly, the proliferation of computing devices beyond the desktop computer. These include hundreds of devices with different sizes and shapes interconnected by wireless communication. Secondly, the physical environment as a main part of his approach, since the user activity is not limited to work in front of a desktop computer. And lastly, the seamless interaction between user and computing devices, doing computers invisible to users. Nowadays, these three points have been summarized into the challenge that users can demand computation capabilities everywhere and anytime.

Ubiquitous computing, also-called pervasive computing, has appeared as a new research branch for mobile computing and distributed systems, and, it has raised new opportunities and challenges in computer science [2]. From a hardware point of view, wireless technologies, processing capabilities and the storage capacity are some of the responsible actors to do computing more pervasive [3, 4]. Original approaches in operating systems, file systems and middlewares have been developed, novel user interfaces paradigms [5] applied, and new application models proposed [6].

[1] "Ubiquitous computing enhances computer use by making many computers available throughout the physical environment, while making them effectively invisible to the user."

R. Meersman, Z. Tari (Eds.): CoopIS/DOA/ODBASE 2004, LNCS 3290, pp. 477–491, 2004.

Moreover, intelligent environments and context-aware computing have run in parallel with ubiquitous computing (see Background section). The first ones provide a framework to support ubiquitous computing applications. The second ones have demonstrated the important role that context plays in ubiquitous computing. Context-awareness and intelligent environment initiatives merge in the current Ambient Intelligence paradigm.

Our work focuses on intelligent home environments. It aims to lead to better computing device interoperatibility. We believe that a global view of the world, shared by every computing device, is necessary to reach efficient device cooperation. Context information guides the structure of this model and provides a better understanding of the relevant information and its relationships. This context model is built from the contributions of every component, and it is dynamically modified as new components appear and disappear.

This paper is organized as follows: first, background work on context and intelligent environments is described. In the next two sections, a context model and a context layer are proposed. After, the results of the previous sections are reflected in a smart room prototype and several context-aware applications. Finally, the future work and the conclusions are explained.

2 Background

2.1 Context and Context-Aware Application

Context has been tied to ubiquitous computing, although the term has had several meanings that differ subtly. The first definitions of context consisted of a list of properties that applications had to be aware of. Schilit [7] highlights that three important aspects of context are: where you are, who you are with, and what resources are nearby.

Pascoe [8] states that "context is a subjective concept that is defined by the entity that perceives it". Thus, context can be any information, depending on the interest of a particular entity. Winograd [9] reinforces the previous statement asserting that "something is context because of the way it is used in interpretation, not due to its inherent properties".

Dey [10] defines context as any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction among a user and an application, including themselves.

Recently, Coutaz and Rey [11] propose an operational definition that relates context to a user involved in a particular task, where context is a composition of a variable state vector over a period of time. The importance of the relationships between the context information is revealed by Henricksen et al [12]. In addition, there are several groups researching in modelling context as a semantic web, such as the Cobra project [13], Aire project [14] and the initiative pervasive semantic web [15].

2.2 Intelligent Environments

An intelligent environment consists of an infrastructure shared by applications, devices and people constrained by physical boundaries. Intelligent environments bring computation into physical world [16]. They are common places where smart devices can interact in a meaningful way.

Research on home automation has focused on hiding computational devices and providing transparent interaction to accommodate to non-technical users. A leading project is The Aware Home [17] from the Future Computing Environments group at Georgia Tech. A real smart house designed to assist elderly people.

There is a great interest in having unencumbered and non-invasive interfaces. One of the first works in this area was the Intelligent Room [18] from the Artificial Intelligence group of the MIT. This room, also-called HAL, consisted of a highly interactive environment which uses multimodal interfaces and embedded computation to allow people to interact with the environment in a natural way. Recently, this work is going on in the Project Aire from the same group [19]. Other related projects are Interactive Workspaces [20], Roomware [21] and SmartOffice [22].

Industry has also shown its interest in this area. The Microsoft Research Vision Group is developing the basic technology to build intelligent environments. The result of this work is EasyLiving [23].

2.3 Our Proposal

According to the previous sections, developing a ubiquitous computing system is a task that should take into account numerous topics. We have centered in two issues:

- To accomplish a seamless integration among pervasive components. There are different and heterogeneous technologies [24]. Moreover, the environment configuration is highly dynamic.
- To obtain a natural interaction that allows to deploy these systems into everyday spaces. User interaction has to keep as flexible as possible. Besides, user preferences and capabilities can change over time and the environment response should adapt to these changes.

As we have pointed out, this heterogeneous mix of software and hardware entities imposes some requirements. In agreement with other works [9, 22, 25] we believe that a global "world model" combined with an asynchronous communication mechanism, is the best approach to achieve complex interactions among components. We propose a context model as a world model (see section 3) and a blackboard architecture [26] (see section 4) as context repository and communication mechanism. Our blackboard implementation differs from other architectures based on tuples where receivers find the information making a pattern-matching mechanism (as Linda [27] or IBM TSpace [28]). In our case, information is stored in a relationship graph, and it is retrieved after traversing through it, as we will show below.

The blackboard allows communicating context changes, finding available resources, and revealing if an entity is added or removed. Information from the blackboard is used by pervasive devices to understand the context and adapt to it. For example, the people in the room and the status of several physical devices (lights,

heating, speakers, etc.) are represented in the context layer, and used to automatically generate a spoken-dialogue interface [29].

3 Context Model

Context is a fundamental part in human communication [9]. This way, it should incorporate into the design of computer systems if we want the human-computer interaction to be more human than computer-like. We propose a context-centric approach that deals with context information representation and distribution. We focus on what is the relevant information that the applications require, without considering how the context is obtained and processed. This facilitates the integration of new components. Next, we will determine what context is and how it can be modelled, and finally, we will describe its distribution mechanism.

As we have seen above, information does not present intrinsic features that allow us to define it as context, but it acquires this category depending on how applications interpret it. In other words, information is transformed into context when it is used. So, any information, independently what it represents, can be understood by an entity as context. According to this, a context model should include all the possible information. Obviously, there is no model that can embrace this complexity. This makes that context models focus on those features which have more probabilities to be required by context-aware applications. Nevertheless, the model should also provide flexible mechanisms to incorporate new information that can become relevant.

The model building has been divided in two steps: firstly, we shall determine on which entities we will acquire context information. People, places, objects, applications and devices are the most common. Secondly, we shall decide which properties of these entities will be measured. As the background section shows, there are several approaches to find the type of information that is frequently used as context. We focus on Dey's two-tiered categorization [10]. He distinguishes between primary and secondary context types.

3.1 Primary Context Type

We have adopted Dey's approach to develop our intelligent environment context model. There are three different main entities. The first one is the place, given that the concept of physical space plays a central role in smart environment. The second one is the person, as the final user of the system. And the last one is the resource, which comprises both physical devices and applications.

Depending on each entity, the primary context varies. For instance, the context of a room is determined by its environmental variables (lighting, noise level, temperature …), of a person by her/his location, identity and activity, and of a resource by its location, handler and state. In order to represent the previous context information, we have distinguished internal context from external context. On the one hand, internal context describes stand-alone properties, on the other hand external context models relationships among context information sources.

Thus, location is represented as a bidirectional relation among a place and a user. This relation is defined in both directions. This way, to ask for who is inside a room is as easy as to know the room where a person is located. For the same reason, relations among places and resources are established. In contrast, mobile resources, such as PDAs or laptops, are not directly tied to a room, but they are related to the person who wears them. In order to know which resources are being used on a certain time, a relationship between the resource and its handler is defined. This handler can be a user or a resource.

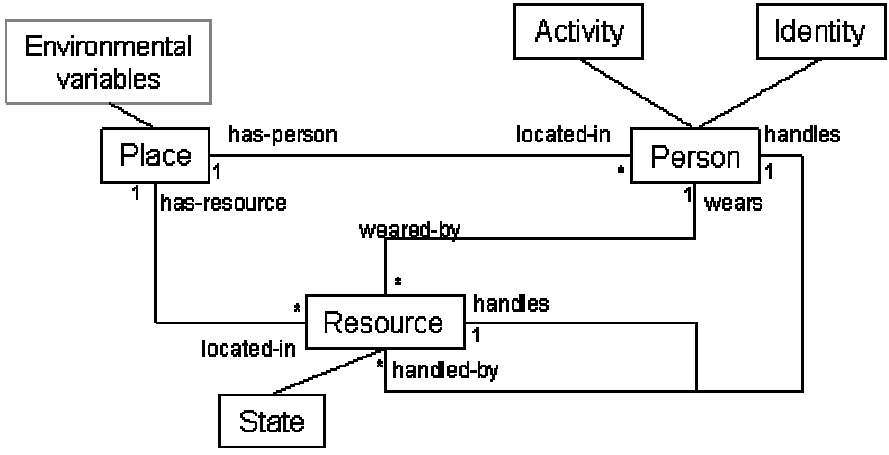

Fig. 1. Primary context relationships and properties

Figure 1 schematically shows the primary context relationships and properties. Notice that context changes dynamically along time, so it is not explicitly included as a part of the model.

3.2 Secondary Context Type

Secondary context types are related with useful but not so frequently used information. This information extends the model described above adding new properties and new relationships. Besides, new entities can be included. This information depends on the domain of context-aware applications. For instance, a contextual audio player application could require that songs would be an entity of the model. This application could necessitate the user's list of favourite songs and which type of song fits with each activity. Then, when a random play is requested, the model information helps to decide which song will be the next.

Our model leads to a semantic network where primary context is the main part. As we will describe, secondary context information is accessible (see namespace section) from primary context information entities since these entities are implemented as indices to any other model information.

4 Context Layer

We have presented the basic context model of our intelligent environment. This section deals with how context-aware applications benefit from it. We propose a middleware, also-called context layer, which allows to notify changes in the context model, discover new context information sources and add them to the model. The context layer implementation lies on a global data structure, called blackboard [26]. This blackboard is a model of the world, where all the prominent information related to the environment is stored. The context layer provides an asynchronous mechanism where senders publish context information in the blackboard and receivers can subscribe to information changes or pull them directly from the blackboard. The published information can be a change in a context property (a door is open), or an entity that has been added or removed (somebody has come into the room). This mechanism permits a loosely-couple among senders and receivers, since it is not needed that both participants are active at the same time or know each other.

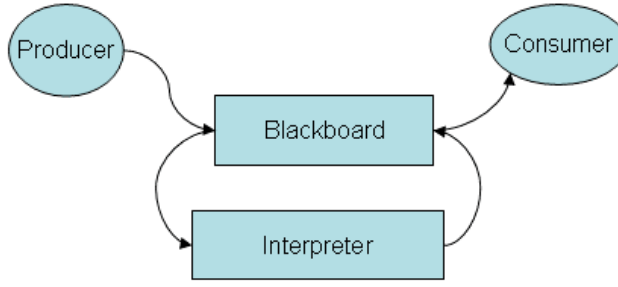

Fig. 2. Interaction between main components of the context layer

Figure 2 illustrates a generic interaction between the main components of the context layer. Producers publish the information gathered from context information sources. Interpreters refine this information and leave it again in the blackboard, and final consumers recover it. Producers measure context directly from real world, providing high-resolution information but with a poor level of abstraction. Interpreters make good this lack by deducing new context properties and relations. Finally, consumers are the context-aware applications.

Components of very different kinds can be found within an intelligent environment. They can be very close to the physical world like sensors, switches, appliances, screens, microphones, speakers, etc. Or they can be related to any kind of software components, such as dialogue managers, intelligent agents, user interfaces, etc.

4.1 Context Representation

Our main goal is to find a structure that can represent not only the relationships among primary types of context but also their properties. The model should also be easily extensible. For these reasons, an undirected graph structure has been chosen to

represent context information. This graph is a data structure composed by a set of nodes and a set of edges. There are two types of nodes: the first one represents an entity and is defined by a name, a type (room, person, resource …) and a list of properties. The second one represents a property. Each property is a name-value pair, where a value can be a literal or another property. There are also two types of edges, those that correspond to the relationships among entities, and those that link entities and properties. It is guaranteed that relations only exist among entities. Thus, the blackboard is composed by an entity graph, where each entity is a tree of properties. This graph is stored in the blackboard and represents a snapshot of the environment context at any moment.

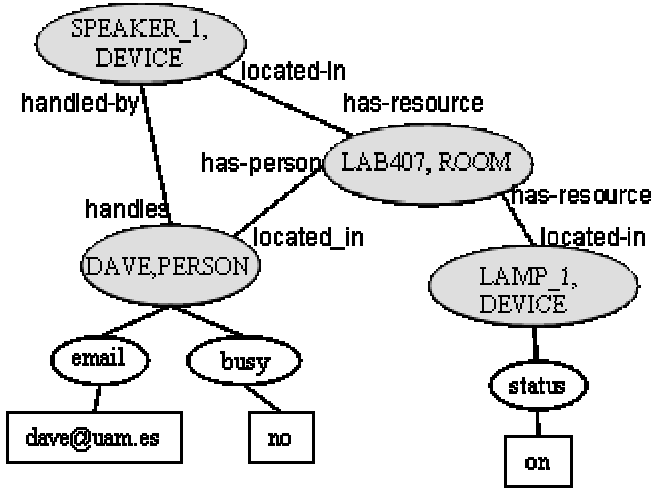

Fig. 3. Simplified blackboard graph

4.2 Name Space

Any node can be located, starting from any entity node and following the relationship path. This is called the node path. It is composed by a list of tokens separated by the slash character. Their order is determined as follows: the first token of the path is the word "name", the second one must be the entity name and the next tokens come as the result of concatenating the names of all the intermediate nodes until the target node. For instance, in the example showed in the figure 3, the lamp_1 status path is */name/lamp_1/status.* In addition, wildcards can be used to substitute one or several tokens. This allows referencing several nodes at the same time. For example, based on the Figure 1, */name/dave/** references all the properties and related entities of the entity *Dave*. As a result it gets the following list: the e-mail and busy property nodes and the Lab_407 and Speaker_1 entity nodes.

Two naming mechanisms are provided to improve the use of wildcards:

– Predefined hierarchy. This mechanism restricts the nodes that compose a path. It specifies how to go through the graph. To do this, each hierarchy defines a sequence of types of entities. For example, the first type of entity must be a room,

the second one a resource, etc… Therefore, when a wildcard is used, only the nodes that match with the expected type will be substituted. These hierarchies are called predefined because they are hard-wired. There is one of these hierarchies for each relationship between primary context entities (room-device, device-room, room-person, person-room, and so on). Following with the example of the figure 3, the path */roomdevice/lab407/*/status* is interpreted as follows: the initial token identifies the hierarchy *roomdevice*. This hierarchy establishes that the first type of entity must be a room followed by a resource. The other nodes remain unrestricted. Therefore, this path references the value of the status of all the devices located in lab407.

– Typed hierarchy. This is a particular case of the previous mechanism. By default, there will be as many hierarchies as types of entities. The initial token of these hierarchies is the type of entity. For example, in the figure 3 there are three default hierarchies: person, room and resource, so that */person/*/mail* retrieves the e-mails from everybody.

4.3 Context Communication

The communication among context producers and consumers is based on a three layered architecture, formed by a physical layer, a context layer and an application layer. The physical layer is related to components that provide properties directly measured from the physical world, while the application layer hosts intelligent agents that deal with properties deduced from the physical world or properties related to the software components.

In our approach the generated context information is published in a central repository accessible to the whole system, following the classical blackboard paradigm.

The blackboard provides standard procedures to request or modify node values, and to subscribe to context changes. Context agents can easily access to the properties of the entities in a transparent way. For example, one property of the context of a room may be the number of people in the room. Several sensors may be used to deduce such information. However, a single final value of the property is produced, and all the other devices and computational entities in the smart environment can use this information independently of the nature of the source.

The interaction process can be summarized as follows. Context producers (or interpreters) send their context changes to the blackboard, and consequently the blackboard modifies the context nodes. Context consumers (or interpreters) notice these changes either by polling the blackboard or by subscribing to blackboard changes. Thus, the blackboard acts as an intermediary, holding the context modifications. The components of the other two layers are responsible to process these changes and to react consequently.

Properties whose values are directly measured from a physical device are managed in a special way. In these cases, the property value is not stored in the blackboard: instead, the blackboard acts as a proxy. Whenever the value is requested, the blackboard asks to the physical device.

In addition to the above behavior, the blackboard provides a mechanism to add and remove relationships between entities.

Besides, the blackboard supports attaching and detaching new entities. When attaching an entity, its representation is sent to the blackboard and stored. The entity relationships must be established in separated operations. If detaching an entity, it and all its relations are automatically deleted from the blackboard. Consumer applications can subscribe to adding and removing relations and attaching and detaching entity events.

Finally, combining all these mechanisms, the required interaction is achieved. For example, if somebody enters an empty smart room, the presence agent notifies this event by adding an entity representing the new person, and establishing a relation among the room and the entity. A context-aware agent can be subscribed to this event and check the value of another node that indicates the current environment luminosity. If it is too dark, then the agent changes the node value that increases the intensity of the lights. Then the physical layer component reacts doing that the lights adjust to the required new state. The reciprocal operation is produced when the person leaves the room.

4.4 Command Heap

In the same way as Johanson and Fox [30] present an event heap to coordinate the interactions of applications, we propose a similar mechanism called command heap to manage conflict resolution. Command heaps are necessary when two or more applications want to change the same part of the blackboard model. For instance, two applications sending contradictory commands about the state of the lights. A command heap is a pre-emptive prioritized command queue. A command represents the desire of an application to change the value of a property or a relation in the blackboard. Each command is composed by an identifier, a priority, an expiration time and a sender's identification. Whenever a command arrives, it is stored in a command heap. If there is no command with higher priority, it will become the active command. Otherwise, it will be placed in the corresponding position of the heap. When a command is activated, the blackboard forwards it to the corresponding application. A sender can delete all its commands or all the commands whose identifiers match with a particular identifier. A command will remain active at the top of heap while its expiration time is valid. If this time expires or the sender explicitly removes the command from the heap, then the next highest priority command will become the active command. The expiration time is limited by an upper bound to avoid blocking a heap for too long. If the application needs more, it may resend the command. Finally, the command priority is chosen by each application and varies in a range of values.

4.5 Blackboard Implementation

Every blackboard is a server that can be accessed using client-server TCP/IP protocols. HTTP has been chosen as the transport protocol because it is simple and widely spread. To exchange information between the applications and a blackboard server an XML-compliant language is employed.

A blackboard provides, at least, the following basic operations:

- GetContext. The client supplies a node path and the blackboard, starting from this node, goes through the graph. For each node its value is obtained, either from the value stored in the blackboard or from a value requested to a physical device. The final result is a tree that is sent to the client. If some values are not available, the corresponding node is left empty, but the rest of the building process continues. Additionally, it is possible to use wildcards to get more than one entity at the same time.
- SetContext: the blackboard receives an order containing a node path pointing to a property and the desired changes. The order is stored in the order heap until it becomes active. Then, the new order is picked up and the value of the corresponding blackboard node is changed. This action may imply modifications outside the blackboard (in other blackboards, or in physical devices).
- SubscribeContext: for each node and each relation, the blackboard stores a list of its subscribed clients. Whenever a node changes, these clients are informed.
- UnsubscribeContext: a client requests that a subscription is cancelled.
- AddContext: this operation allows to dynamically add an XML representation of a new entity to the blackboard.
- RemoveContext: removes the referenced entities and the relationships associated to them.
- AddRelationship: this order establishes a relationship between two entities. It will be effective when the order is active.
- RemoveRelationship: the opposite order to the previously described.

Moreover, blackboard designers are provided with a tool that assists them in the construction of the blackboard. A compiler has been developed which produces a blackboard implementation from an XML file. This file specifies the node names, their initial values and their hierarchical structure. Finally, there is also an additional tool that processes the XML file to obtain comprehensible documentation.

4.6 Information Flows

We also find suitable the use of relationships to represent the flow of information among physical devices. The most interesting case is modelling how multimedia data (image, audio and video) flows through a room. For each multimedia resource, an entity is defined in the blackboard. When two multimedia resources have to be connected the corresponding relation is added to the blackboard. Then, both resources configure themselves to satisfy the new situation. A similar behaviour occurs when the relation is removed from the blackboard and the flow of multimedia information stops.

As an example, we have developed a context-aware application that changes the pictures showed in several flat-screens depending on the current occupants of the room. Following the interaction model depicts at figure 2, the application is decomposed in three modules: an image source, that acts as a producer, a manager, which plays the role of an interpreter, and one or more images sinks or consumers.

4.6.1 Image Source

This module decides which picture has to be displayed at each moment. A list of the room occupants is stored in a circular queue. Besides, for each occupant, there is another circular queue that holds the URLs of his or her favourite pictures. The URLs of the pictures that will be displayed are chosen following a two-phase selection procedure. Firstly, an occupant is selected from the occupant's queue, and secondly, an URL is picked up from his or her favourite picture URL queue. Both queues use a round-robin algorithm to select the next candidate. The selected URL is stored in the blackboard and broadcasted to every related sink.

Whenever a new person enters the room, the module is notified. This person is added to the occupant's queue, and a new circular queue is created to store his or her list of URLs. When an occupant leaves the room, his or her favourite picture URL queue is deleted, and the person is removed from the occupant's queue. If nobody is inside the room, a default queue storing two pictures is used.

At start up, the image source module stores in the blackboard: the URL of the first selected picture and the time that the selected URL remains valid.

4.6.2 Image Sink

Each image sink module manages a flat-screen. The image sink is idle until a relationship is established between an image source and itself. Then, the image sink consults the blackboard and retrieves the URL of the selected picture. The sink requests the picture using the HTTP protocol and displays it on the screen. This process is repeated whenever the image source selects a new URL, and keeps on until the relationship is removed from the blackboard.

4.6.3 Manager

A manager is any application capable of adding and removing relationships from the blackboard. Managers decide which sources are connected to which sinks creating a dynamically updated network of connections. The lists of sources and sinks are available in the blackboard and the manager reads them to set up the interface. The lists are updated when sources or sinks appear or disappear. The manager can also configure the refresh time of the sources.

We have developed a graphical interface tool that allows users to manually configure the connection network between sources and sinks. This tool can be easily adapted to manage other types of multimedia traffic, such as audio and video.

5 A Smart Room Prototype

A laboratory has been transformed into two rooms. The main room is equipped like the living room of a typical house and the adjacent one is equipped like an office. The context layer described above harmonizes the interaction between the components of these rooms. The laboratory is composed by a set of heterogeneous devices and applications. The context graph includes the representation of the installed devices and the associations among them.

Two physical networks have been deployed. For the connection of sensors (presence, temperature, luminosity, etc.) and actuators (switches, engine controllers, etc.) we utilize the European bus EIB. This bus tries to set a standard within the European Union for home automation. For the multimedia information flow (images, digital radio, ip-camara, etc.), we are using an Ethernet network. Each device can be connected to either network (or both), depending on its nature, and has access to the context blackboard through them. The access to the physical layer is uniformed by a SMNP (Simple Management Network Protocol) layer. This is described in [31].

The installed devices can be divided in three categories:

- **Home automation.** Composed of several independent systems: an automatic lock used to control the physical access, photoelectric sensors that inform when someone enters the room, a smart card system that identifies the users and several EIB devices, such as room lights, switching devices, an alphanumeric display, etc.
- **Audio-Video information.** It includes a digital radio, a TV set, two hi-fi speakers, a DVD player and several flat monitors that can be used alternatively as output devices for video or as system interfaces.
- **Voice interaction.** Wireless microphones that provide the users with free-movements.

We have developed several demonstrators that range from simple proof-of-concepts to release applications. Our purpose is to develop each demonstrator independently from the others. Furthermore, these demonstrators do not have to know either how the context information is generated or which context producers are involved.

These applications are grouped into three categories. The first two categories focus on different kinds of context changes. The first one deals with changes on context properties, while the second one studies the potential of the model of relationships. The third category includes two user interfaces that employ the context to customize their functionality. The three categories are:

- **Access applications**. They are interested in changes on the state of the main gate. There are two applications of this kind: the first one sends an e-mail to the room owner when the door is open for a long time and, if someone is inside the room, utilizes the voice synthesizer to notify him or her. The other prevents intruders. If an unauthorized person enters the room, it triggers a chain of events: the room lights turn on (if they were off), a web-cam takes a picture which is sent to the room owner via e-mail and, finally, an acoustic alarm goes off.
- **Person-identification applications**. They focus on services that depend on the identity of the people inside the room. Every time a relation between the room and a person is added or removed, these applications are notified. Besides, the number of people and their identifications can be retrieved from the blackboard at any time. We have developed several applications of this type: (a) The contextual picture application described at the Information Flow section. (b) A meeting-aware application. When it determines that a meeting is taking place it sends an e-mail to the rest of possible attendants. (c) A speech application that utilizes a voice synthesizer to make custom greetings when a user enters or leaves the room. The salutation is adapted to the user and to the time of the day. (d) A simple illumination module that turns the light on when the first user enters, and turns it off if nobody is inside it.

– **User interfaces**. Finally, two independent user interfaces have been integrated into the smart room: a web-based user interface [32] that permits to control the devices of the room and a spoken natural language dialogue system [33] that permits the user to interact with the environment. Both of them are dynamically set up using the blackboard information.

6 Current and Future Work

Our model relies on set of blackboard servers. Each server is associated to one environment, and provides a set of services to its computational devices. The representation of the mobile entities is attached or detached to the blackboard as they are carried in and out the rooms. The current blackboard implementation compels to send all the information related to the mobile entities. We are improving the current mechanism in order to allow sending a link that points to where the information is stored. This link is treated as another relationship, so that applications will continue perceiving a global view although its implementation is distributed.

The other research line, which is being explored, deals with how to apply semantic web technologies to our prototype. As we have explained in background section, there are several research groups that are integrating semantic web into pervasive computing, and they have obtained fruitful results in this area. Our work will aim to translate our current XML-compliant representation language to RDF or OWL. These languages exhibit interesting features that improve the representation model of entities and their relationships. Following this approach, we are developing a smart home ontology where the primary context model is refined and domain-dependent context information is added.

Finally, we are carrying on the implementation of a contextual broadcast audio player. This application uses the blackboard information to find out where the user is located and which speakers are available. This way, the sound can follow the user from room to room. The desired noise level of the user will be taken into account, as same as the preferences of another user in the room.

7 Conclusions

The present work addresses the interaction between ubiquitous computing devices. We have considered intelligent environments as a particular case of ubiquitous computing applications, and we have chosen a home environment as our framework. The problems that arise in an intelligent environment have been studied. In particular, those related to the deployment of heterogeneous technologies and the achievement of a natural user interaction. A common factor of these problems is that environment and its components produce highly dynamic context information.

We have proposed a context layer as the glue to achieve the required synergy among pervasive computing devices in order to constitute a smart environment. This context layer is based on a unified model view of the world shared by every computing devices and accessible using an asynchronous communication mechanism. This relies on a data-centric approach where the main goal is to publish the changes

on the context in a common and structured repository independently of the source and how they are generated. Besides, a single interface, which abstracts from the communication details of the various computing devices, is also provided. An order heap is employed to solve conflicts between components that exchange the same information.

The context information is stored as graph and this is structured following a proposed context model. We have proposed that context can be represented by internal properties and by external relations between entities. Besides, we have followed Dey's approach to distinguish among primary and secondary context types. This classification guides the implementation of the model and aids developer and applications to find out the context information in the blackboard.

Several applications have implemented to demonstrate the utility of the relationships to model the context. In particular, we have explained how we utilize them to manage the flow information. Moreover, we have successfully developed two user interfaces that exploit the blackboard advantages. All of these applications have been tested in a real environment.

Acknowledgement. This paper has been funded by the Spanish Ministry of Science and Education, project number TIN2004-03140.

References

1. Weiser, M.: Some computer science issues in Ubiquitous Computing. Communications of ACM, 36(7). July (1993) 75-84
2. Satyanarayanan, M.: Pervasive Computing: Vision and Challenges. IEEE Personal Communications, 8(4). August (2001) 10-17
3. Want, R., Borriello, G., Pering, T., Farkas, K. I.: Disappering Hardware. IEEE Pervasive Computing, 1(1). January-March (2002) 36-47
4. Schilit, B. N.: Mega-Utilities Drive Invisible Technologies. IEEE Computer, 36(2). February (2003) 97-99
5. Pingali, G., Pinhanez, C., Levas, A., Kjeldsen, R., Podlaseck, M., Chen, H., Sukaviriya, N.: Steerable Interfaces for Pervasive Computing Spaces. In Proceedings of PerCom'03. March (2003)
6. Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J., Zukowsk, D.: Challenges: An Application Model for Pervasive Computing. In Proceedings of Mobicom 2000. (2000) 266-274
7. Schilit, B., Adams, N., Want, R.: Context-Aware Computing Applications. IEEE Workshop on Mobile Computing Systems and Applications. (1994)
8. Pascoe, J.: Adding Generic Contextual Capabilities to Wearable Computers. In Proceedings of 2nd International Symposium on Wearable Computers. (1998) 92-99
9. Winograd, T.: Architectures for Context. Human-Computer Interaction, 16(2,3& 4). Lawrence Erlbaum Associates (2001) 401-419
10. Dey, A.: Understanding and using context. Personal and Ubiquitous Computing, 5(1) (2001)
11. Coutaz, J., Rey, G.: Foundations for a theory of contextors. Computer-Aided Design of User Interfaces III. Kluwer Academic Publishers. May (2002) 13-34

12. Henricksen, K., Indulska, J., Rakotonirainy, J.: Modeling Context Information in Pervasive Computing Systems. Pervasive Computing, First International Conference, Pervasive 2002. LNCS 2414. Springer-Verlag (2002) 167-180
13. Chen, H., Finin, T., Joshi, A.: Semantic Web in in the Context Broker Architecture. In Proceedings of PerCom 2004. March (2004)
14. Peters, S., Shrobe, H.: Using Semantic Network for Knowledge Representation in an Intelligent Environment. In Proceedings of PerCom'03. March (2003)
15. http://pervasive.semanticweb.org
16. Coen, M.: Design Principles for Intelligent Environments. Proceedings of the AAAI Spring Symposium on Intelligent Environments. (1998)
17. Kidd, C. K., Orr, R., Abowd, G. D., Atkenson, C. G., Essa, I. A., MacIntyre, B., Mynatt, E., Starner, T. E., Newstetter, W.: The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In Proceedings of the Second International Workshop on Cooperative Buildings, CoBuild'99 (1999)
18. Coen, M.: Building Brains for Intelligents Environments. In Proceedings of the 1998 National Conference on Artificial Intelligence (AAAI98) (1998)
19. http://www.ai.mit.edu/projects/aire/
20. Johanson, B., Fox, A., Winograd, T.: The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. IEEE Pervasive Computing Magazine, 1(2). April-June (2002) 67 – 74
21. Tandler, P., Streitz, N. A., Prante, T.: Roomware-Moving Toward Ubiquitous Computers. IEEE Micro, 22(6). November-December (2002) 36-47
22. Le Gal, C., Martin, J., Lux, A., Crowley, J. L: SmartOffice: Design of an Intelligent Environment. IEEE Intelligent Systems, 16(4). July-August (2001) 60-66
23. Brumitt, B.L., Meyers, B., Krumm, J., Kern, A., Shafer, S. A.: EasyLiving: Technologies for Intelligent Environments. In Proceedings of Handheld and Ubiquitous Computing, 2nd Intl. Symposium. (2000) 12-27
24. Haya, P., Alamán, X., Montoro, G.: A Comparative Study of Communication Infrastructures for the Implementation of Ubiquitous Computing. UPGRADE 2(5) (2001)
25. Maglio, P., Campbell C.: Attentive Agents. Communications of the ACM, 46(3). July (2003) 47-51
26. Engelmore, R., Morgan, T.: Blackboard Systems. Addison-Wesley (1998)
27. Gelernter, D.: Generative communication in Linda. ACM Transactions on Programming Languages and Systems, 7(1). January (1985) 80-112
28. Wyckoff, P., McLaughry, S., Lehman, T., Ford, D.: TSpaces. IBM Systems Journal, 37(3) (1998) 454-474
29. Montoro, G., Alamán, X., Haya, P.: A plug and play spoken dialogue interface for smart environments. In Proceedings of CICLing'04. LNCS, Vol. 2945 (2004)
30. Johanson, B., Fox. A.: The EventHeap: A Coordination Infrastructure for Interactive Workspaces. In Proceedings of WMCSA 2002. (2002)
31. Martinez, A.E., Cabello, R., Gómez, F.J., Martínez, J.: Interact-DDM: A Solution for the Integration of Domestic Devices on Network Management Platforms. In Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (2003)
32. Alamán, X., Cabello, R., Gómez-Arriba, F., Haya, P., Martínez, A., Martínez, J., Montoro, G.: Using context information to generate dynamic user interfaces. In Proceedings of 10th HCI International Conference (2003)
33. Montoro, G., Alamán, X., Haya, P.: Spoken interaction in intelligent environments: a working system. Advances in Pervasive Computing. (2004)