Ubiquitous Developments in Ambient Computing and Intelligence:

Human-Centered Applications

Kevin Curran
University of Ulster, Northern Ireland

Information Science
REFERENCE

Senior Editorial Director: Kristin Klinger
Director of Book Publications: Julia Mosemann
Editorial Director: Lindsay Johnston
Acquisitions Editor: Erika Carter
Development Editor: Michael Killian
Production Coordinator: Jamie Snavely
Typesetters: Jennifer Romanchak & Michael Brehm
Cover Design: Nick Newcomer

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com/reference

Library of Congress Cataloging-in-Publication Data

Ubiquitous developments in ambient computing and intelligence: human-centered applications / Kevin Curran, editor.
p. cm.
Includes bibliographical references and index.
Summary: “This book provides a comprehensive collection of knowledge in cutting-edge research in fields as diverse as distributed computing, human computer interaction, ubiquitous computing, embedded systems, and other interdisciplinary areas which all contribute to ambient computing and intelligence”--Provided by publisher.
ISBN 978-1-60960-549-0 (hardcover) -- ISBN 978-1-60960-550-6 (ebook) 1. Ubiquitous computing. 2. Ambient intelligence. 3. Electronic data processing--Distributed processing. I. Curran, Kevin, 1969-
QA76.5915.U264 2011
004--dc22

2011009954

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 20
Distributed Schema-Based Middleware for Ambient Intelligence Environments

Javier Gómez
Universidad Autónoma de Madrid, Spain

Germán Montoro
Universidad Autónoma de Madrid, Spain

Pablo Haya
Universidad Autónoma de Madrid, Spain

Manuel García-Herranz
Universidad Autónoma de Madrid, Spain

Xavier Alamán
Universidad Autónoma de Madrid, Spain

ABSTRACT

In this work we present a middleware developed for Ambient Intelligence environments. The proposed model is based on the blackboard metaphor, which is logically centralized but physically distributed. Although it is based on a data-oriented model, some extra services have been added to this middle layer to improve the functionality of the modules that employ it. The system has been developed and tested in a real Ambient Intelligence environment.

INTRODUCTION

The Ubiquitous Computing term was coined by Mark Weiser in 1991 (Weiser, 1991). From that moment on, many problems and opportunities have arisen from that vision of a world rich in information and interaction. Ambient intelligence environments (also called intelligent environments) are one of the fields where Ubiquitous Computing can be naturally applied. We can define an active environment as a space limited by physical barriers, which is capable to sense and interact with its inhabitants.

DOI: 10.4018/978-1-60960-549-0.ch020

The definition leads to the necessity of some kind of physical infrastructure for sensing and acting into the real world. However, as we will show below, these environments present some particular problems beyond hardware issues. For instance, the environment configuration changes dynamically and client applications should be notified of these changes. Thus, a software infrastructure is also needed to solve these problems.

The approach that we present in this work tries to solve these issues, making easier the developing task and the interaction among applications. For this, it employs a common, normalized and formalized definition of the reality. This definition, and the information that it stores, should be accessible and shared by clients and applications.

Moreover, some extra features have been added to the system to provide additional services, such as an historical registry, which shows all the activity carried out by the system or a rule-based service, which changes the behavior of the environment under some circumstances.

Another interesting feature is one that adds a description of the representation of the elements that compose the environment. This feature facilitates the definition and development of interfaces to interact with the environment. User Interfaces are becoming an important subject in the Ambient Intelligence field, because computers usually keep hidden from users and system services are obtained by means of context awareness interaction. Moreover, this interaction must be adapted to the task, the environment, its occupants and the available resources (Paterno & Santoro, 2002; Rayner et al., 2001). The integration of this description with the rest of the elements of the model helps to fulfill this task.

Finally, as an important aspect of our development, this model and its services have been tested in a real intelligent environment.

MOTIVATION

Any intelligent environment is composed by a heterogeneous set of software and hardware components (Haya, et al. 2001). This involves some challenges:

- **Bounded environment**. Human activities are usually taking place in a discrete and bounded environment. As Kindberg and Fox (2002) pose in their *boundary principle*, designers should be aware of this distribution. In this respect, each smart space partitions the whole domain in isolated management areas. That is, in a house environment, in example, management resource policies are spatially limited to the home extension, and homeowners should decide them. This is particularly true for privacy concerns since humans consider home as a private space and they would like to manage it following their own criterions.
- **Heterogeneous components**. Smart homes are populated by a heterogeneous set of numerous components that can be either software or hardware. So it is needed to integrate and manage different kind of technologies. This leads to a more complex development process. This complexity affects to every component of the smart home. In particular, the final user would like to interact with the environment using different modes (such as voice, gestures, tactile, etc.) This implies, practitioners have to deal with very different user interaction techniques. Besides, the distribution of the information required to choose among different communication networks depending on several factors such as bandwidth constraints, mobility or deployment cost.
- **Highly distributed components**. Both sensors, whose task is to catch informa-

tion from the environment, and actuators, which are the devices that make changes in the real world, are located in different places. New extra impediments appear during the developing process of a distributed system. Some particular tasks, such as configuration and debugging, are more complicated in a distributed framework.

- **Dynamic configuration**. Smart home environments are highly dynamic. It cannot be possible to predict when users go in or leave the environment, and also, when devices are attached or detached. The system has to be always running. For instance, home's inhabitants consider inadmissible periodically failures of their homes. This implies that the management of new components has to be done at run-time.

The process of developing applications for intelligent environments requires a software infrastructure to deal with these problems. Programmers require both being able to obtain the information from an individual component, and accessing to the global state of the environment. This also includes information from non-computational elements (users, objects, time, etc.) and the relations between them.

An abstraction layer is required to allow the management of the information relative to the environment. Therefore, we propose an improvement of a previously developed middleware (Haya, et al. 2004) that allows accessing to the information that comes from either hardware or software elements, and that provides them with additional features.

The process of developing applications for intelligent environments requires a software infrastructure to deal with these problems. Programmers require both being able to obtain the information from an individual component, and accessing to the global state of the environment. This also includes information from non-computational elements (users, objects, time, etc.) and the relations between them.

RELATED WORK

Much taxonomy can be found in literature to classify middle layers. On the one hand, attending to its programming model, we can classify them in:

- **Service oriented**: Based on the client – server model. Client applications access to the context by means of a standard communication interface.
- **Data oriented**: They are centered in context representation. The distribution is carried out by a reduced operations set. The mechanism the middle layer uses to distribute information can be classified regarding the spatial and temporal coupling level:
- **Temporarily and spatially coupled**: A process communicates with a known receptor only if they coincide with each other in time.
- **Temporarily coupled and spatially uncoupled**: Process group temporarily to transmit information, but they do not have to know each other.
- **Temporarily uncoupled and spatially coupled**: The emitter process needs to know the receptor(s) of the data it is sending, but they do not have to coincide in time.
- **Temporarily and spatially uncoupled**: Transmitter processes do not know the receptors(s) and they do not have to coincide in time.According to the technology employed in its implementation, we can divide them into:
- **Distributed Objects**: The basic unit is the object (active or passive). Active objects are contextual information sources and can be queried by object remote calls.

- **Infrastructure**: It is based on an infrastructure of known, reliable and public servers that provide a set of services.
- **Blackboard**: It is a centralized mechanism in which context aware applications store and get information back from a common, known and accessible repository. o Finally, regarding to its purpose:
- **Acquisition and processing of context**: They provide mechanisms to standardize the communication with the different technologies. In some situations, they can process some of the data to obtain a more abstract representation
- **Distribution oriented**: They want to get an effective context distribution

Middle Layers for Intelligent Environments

Accord (Akesson, 2000): This tool allows configuring the devices of a house environment in a flexible way. Each device publishes its state in a common shared space. The editor is oriented to configure the environment by the final user.

- **Aware Home** (Kidd, 1999): This project aims to build a house to be used as an ubiquitous computing lab to support life in that house
- **Beach** (Tandler, 2004): (Basic Environment for Active Collaboration) It is a platform developed in SmallTalk that makes the creation of hypermedia collaborative applications easier
- **DOBS** (Villanueva, 2009): Poses a framework oriented to distributed objects to design services. It is composed of a set of modules, as interfaces, audio and video services, common and integration platform services and an information model.
- **EmiLets** (de Ipiña, 2006): Presents a middle layer that tries to facilitate the creation of spaces with intelligent objects and supports the mobile phone as the remote control of all of that devices.
- **Gaia** (Román, 2001): On the one hand, it is based on distributed objects. On the other hand, it similar to a classical distributed operating system. It is composed of a collection of services that provides a programming interface. This way, the environment and the resources that composes it as if they were a unique and programmable entity.
- **ICrafter** (Ponnenkanti, 2001): This system focuses on a flexible services composition. It looks for facilitating the creation of user interfaces from the available services in a moment. A service can be either a device or an application
- **InConcert** (Brumitt, 2000): This is the middle layer that was used in the Easy Living Project (Microsoft Research division). The communication mechanism is asynchronous, XML-based and it uses a machine-independent addressing.
- **IDP** (Choi, 2006): This middle layer provides in-home services based on biometric information and context. The middle layer receives biometric information, such as heart rhythm, facial expressions, body temperature, location and person movements, from the sensors deployed in the environment.
- **Metaglue** (Phillips, 1999): Provides a coordination mechanism for big groups of software agents. Some facilities were added to the system, such as new services discovering, resources acquiring policies, etc. It is an extension of the Java programming language.
- **OSGi** (Gong, 2001): This approach tries to standardize the connections between devices (inside or outside of the house) to facilitate VoIP, TV on demand, remote control services, etc.

- **Plan-B** (Ballesteros, 2006): This work proposes a new approach. Instead of using a middleware, it is based on a virtual file system in which all of the elements of the environment are organized.
- **Semantic Space** (Wang, 2004): This infrastructure for intelligent environments is based in context. There are three key aspects: An explicit knowledge representation (in RDF and OWL), a search engine (based on RDQL) and a reasoner that allows inferring new situations from the information stored in the knowledge base
- **SmartOffice** (Le Gal, 2001): This work presents an integration resource-oriented protocol. Every module communicates with the resource server supervisor.

MIDDLEWARE LAYER: THE BLACKBOARD

The blackboard metaphor poses that all information exchange is done through a logically centralized module, where producers publish their output without knowing who will consume them.

In our case, the blackboard is a physically distributed middle layer between elements of the environment and applications. It presents multiple characteristics, oriented to solve the problems presented previously. It can be studied under three points of view: from a data model point of view, an application model point of view and a communication model point of view.

Data Model

The data model is a representation of the information relative to the world, which is independent from the source that generates it and the abstraction level. It is divided into two clearly different but narrowly related parts: The schema model and the repository. The schema model contains the description of the world, in terms of classes, their properties, capabilities and the relations that can appear between them, that is, an ontological model. On the other hand, the repository stores entities that are the realizations of the classes of the schema. Entities can represent physical objects, such as computers, people, etc. or virtual objects, such as pictures, personal information, songs, etc. Regardless of the nature of the entities, all their information is accessible through the global information structure.

This representation allows combining both abstract concepts and information from sensors. Some of the advantages of this data model are:

- **Functionality and data are separated**. This allows developing each part individually. A data oriented middle layer can be the base of a service oriented middle layer, so the data layer can be reused by several services.
- **Uncoupled communication**. The components of the architecture are more independent thanks to a decoupled communication mechanism. The blackboard model (see forward sections) makes the coordination between applications that interact with the environment easier, since they do not have to be synchronized either temporally or spatially. The fact that two applications do not have to be synchronized temporally involves that to communicate them, it is not necessary to run both at the same time. On the other hand, the lack of space synchronization produces that two applications do not have to know each other to interchange information. This is possible because the blackboard stores all the information of the environment. This kind of communication makes the reconfiguration process more efficient in dynamic environments, since the procedure to follow when a new component appears or disappears is transparently carried out.

- **Multiple and heterogeneous information sources are allowed**. The information of the environment may come from sensors and also be deduced. Following the data-oriented paradigm, we have defined a common schema that establishes the representation of the data stored in the blackboard. This schema has been designed to reflect a universal description of the environment components, independent of any particular technology. In doing so, many different applications can reuse the same information. This implies, on the one hand, that system the functionality emerges from the different use that distinct modules do of the same information and, on the other hand, that producers may improve their acquisition procedures while the data model remains unalterable. For instance, location information is obtained using different levels of abstraction (i.e. room level or spatial coordinates). Consumers choose which level of detail is desired, and subscribe to changes on the pertinent variables. An improvement in the location mechanism does not affect to the consumers, since the measured variables remain the same.
- **Straightforward creation of user interfaces.** Our blackboard approach makes easier to keep only one application model that can be reused in several personalized interfaces. Thus, the MVC (Model-View-Controller) pattern (Reenskaug, 1979) that has been used in the user interface implementation is transferred easily because the separation between the three components arises naturally. Moreover, the information represented in the common schema makes possible to automatically generate different and adapted interfaces (Gómez, et al., 2008).

A data oriented approach does not exclude the middle layer to provide other services. For example, to store an historical register of the contextual information could be an interesting service since lots of applications can demand it.

Application Model

Blackboard architectures are considered a classic paradigm (Engelmore & Morgan, 1988) that has been proved to be used in control systems (Hayes-Roth, 1985). Blackboard architectures were used to solve non-deterministic problems (Erman, et al., 1980). The solution was found by making some modules to cooperate; each one specialized in one specific task. Every process stored partial results in a blackboard. There was also a centralized coordinator whose task was to choose, reject or merge those partial results. Each module only knew the blackboard and did not know any extra information about the rest of the system. So, a data-centered view is considered, instead of one centered in the process. System components do not communicate each other directly, but they sent requests to a central repository. To take information from the system, these components can subscribe to changes on the blackboard or access to it directly. Some characteristics of our architecture are:

- **Common data model.** The information stored in the blackboard follows a common model. As it was said in previous sections, this model separates data and functionality, allows applications to be uncoupled both spatially and temporally and the MVC pattern makes the implementation of interfaces easier.
- **Logically central repository.** The middleware allows accessing all the environment information as it was stored in a unique repository. On the application developer side there is only one blackboard. But it is composed by distributed set of spaces that manage a part of the global information. This solution was proposed to make easier

the implementation task, without losing the scalability of the whole system.

- **Communication mechanism.** There are two types of protocols. On the one hand, there is a polling-based protocol, which allows obtaining the information directly from the blackboard. On the other hand, there is a publish-subscribe protocol, where the sources of information publish on the blackboard changes in the context, and consumers subscribe to these changes to receive them.

Communication Model

As we mentioned above, the collection of devices, people, relations, etc., which establishes the model of the world, is available for developers through the blackboard. Classical implementations are based on a shared tuples spaces (Gelernter, 1985). Nevertheless, our blackboard implementation is based on a directed graph composed by entities and relations. This representation fits better with the organization of the real world than the model based on tuples. Another advantage of this model is that navigation through the instances of the model is easier.

Applications can access the blackboard by means of a set of operations that has been defined to allow asking the blackboard for instances or updating them. It also allows discovering new instances or relations, and subscribing to their changes.

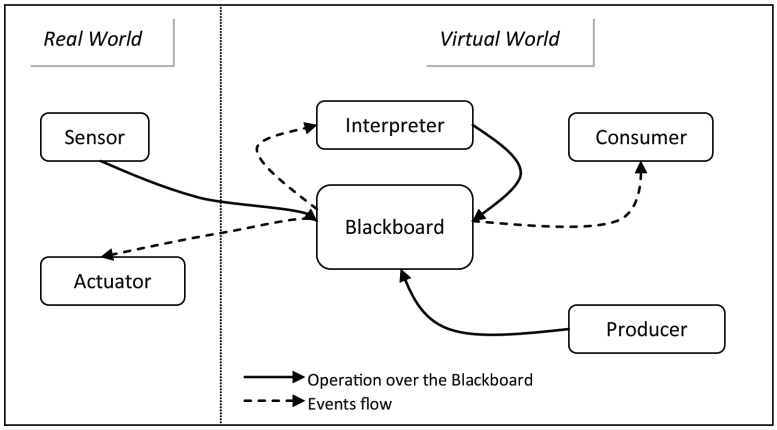
For a better understanding of the communication model we have established a division between the different kinds of modules that can interact with the blackboard. This classification establishes two axes: virtual-real world and publisher-consumer of information. Thus, we distinguish between five types of clients:

- **Sensors.** Information sources linked to the real world are included here. Sensors measure information from the real world directly, in a low level of abstraction.
- **Actuators.** They make changes in the real world from the modifications in the blackboard. As Sensors, they are physical entities but virtually represented in the blackboard.
- **Interpreters.** These components subscribe to changes in the blackboard. They can be classified in two subcategories, depending on how they process the information: They can turn the information from sensors to high-level information or they can divide a complex task in simpler actions.
- **Consumers.** They are the final receptors of the changes produced in the blackboard. User Interfaces and autonomous event-based applications are included here.
- **Producers.** This group is formed by applications that update the model stored in the blackboard. These changes can be received by a Consumer or alter the real world.

Figure 1 summarizes the interaction between these five types of clients. As it is shown, the blackboard acts as a "meeting point" for the components of the system. Arrows in broken line show events generated by changes in the model. These arrows recover information about entities and relations. On the other hand, arrows in solid line show two different operations: look up information or updates. These arrows modify the value of a property or add/remove entities or relations to the model. When producers/sensors/interpreters need to communicate new changes, they modify the information in the blackboard. When consumers/actuators need the information, they can either ask the blackboard directly to see if any modification has happened or subscribe to the modifications and, when a modification happens, be notified.

One of the advantages that the blackboard paradigm provides is that clients do not have to know the existence of the rest of the components;

Figure 1. Communication model used in a blackboard-based architecture



each client only knows about the existence of the blackboard and the part of the model in which it is interested. This approximation let clients be uncoupled either temporally, spatially and functionally.

To be uncoupled spatially and temporally involves that processes do not know the receiver of their messages and do not have to share the same time frame. The communication is carried out using a sharing mechanism in which emitters leave the information that would be gathered by receptors when it is needed.

To be functionally uncoupled means that producers do not have to know how consumers will use the information. These three uncoupling levels mean an advantage in dynamical environments, since the system do not have to be reconfigured after adding/removing components.

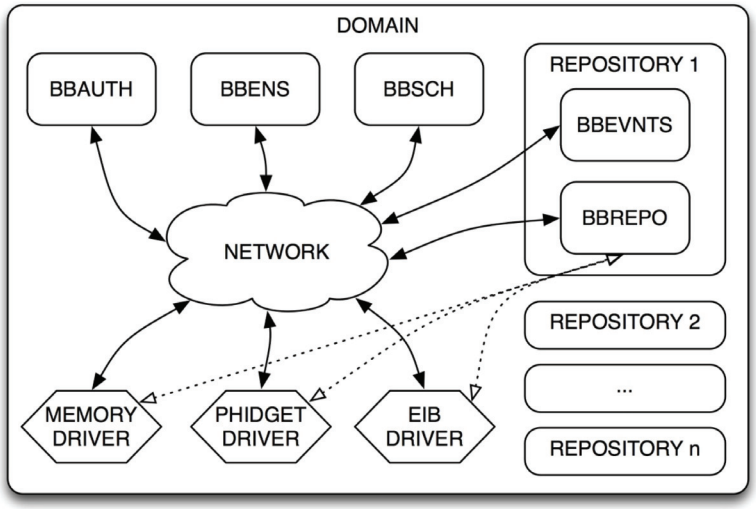
Architecture

We divide the whole space into domains in order to improve scalability. Each domain, as it will be shown below, is composed of a set of servers and drivers. These domains could be, in turn, distributed in a network.

While the schema model is shared by all the elements of the domain, the repository can be divided into parts. There are also additional modules: an authorization server, an entity name solver server and drivers. These drivers are responsible for allowing the correct access to the physical device or even to memory data. A diagram of a domain is shown in Figure 2.

- **BBAUTH**: This server manages user authentication in the blackboard system. It also acts as a DNS that resolves the access to the rest of servers that compose the domain.
- **BBENS**: It is a DNS focused on entities, i.e., it resolves an entity request by returning the information needed to connect to the specific repository that contains the entity.
- **BBSCH**: This is the schema model. It is unique for the whole domain, so all the repositories share the information that this server provides. Whenever a new element is added to the blackboard, the schema server must be checked for model agreement.
- **REPOSITORIES**: There can be more than one repository. They are composed of two servers: the repository itself and another one dedicated to notify applications about changes made over the set of entities of the specific repository to which it is as-

Figure 2. Architecture of the System: Repositories, modules and drivers



sociated (event triggering and the publish/subscribe mechanism will be explained in the next section).

- **DRIVERS**: They are responsible for allowing access to the physical device or to memory-stored data. They connect through the network to the repository they are associated to. We have developed three kind of drivers until now:
  - **Memory**: provides access to memory-stored data
  - **Phidget**: enables access to Phidget devices.
  - **EIB**: allows access to elements that communicate through an EIB Bus.

Basic Operations

We have defined a set of basic operations that support the communication model described above. It can be summarized in the following five operations:

- **Look up or modify properties**. The blackboard provides a standard set of functions that give access to the property. Properties whose values are measurements of physical devices are treated in a special way. This is, the value of the property is not stored in the blackboard. Instead, the blackboard works as an intermediate. When the value is required the blackboard asks directly the device for it.
- **Look up or modify relations.** The blackboard has the capability to add or remove relations between entities. Relations are used as a mechanism to show the connections between the elements of the environment at any moment. For example, relations are used to represent locations or memberships.
- **Look up or modify entities.** Like relations, entities can be added or removed as they become a part or are removed from the environment. When a new entity is added, its representation is inserted in the blackboard. This representation includes the name of the entity, its type, properties and capabilities. If an entity is removed, all

its information is not longer available in the blackboard.

- **Look up or modify capabilities.** Capabilities work in the same way as properties, but the entities can acquire or lose them in execution time. Meanwhile the entities represent an intrinsic characteristic of the entities the capabilities describe functional features. In other words, they represent actions that can be performed by an entity.
- **Subscription to changes.** There is an asynchronous-event mechanism that allows clients to subscribe to changes in blackboard. Subscriptions are useful when a client needs to monitor the value of a part of the data model throughout the time.

With this set of operations, the system provides clients a simple and common mechanism to access the information. They do not have to worry about how the blackboard gets the value from a sensor or deduce it.

Blackboard Services

As it was mentioned before, a data-oriented approach does not exclude a service-oriented approach. There are some basic services that are widely requested by the clients. These have been included as an integrated part of the middleware.

Three of the services developed so far are: a "log service", which keeps an historical registry of all the operations invoked in the blackboard; a "rule service", which specifies common behaviors for indirect control; and an "interface service", which makes easier the labor of developing interfaces for direct control.

Log Service

This extra information source for clients is an historical registry that stores the actions that took place and the actors that took part on them. Thanks to this service the blackboard can answer to questions like "Who turned the light on?" After that, the service checks the log for the last time the light was turned on, and provides something like *"<light> <turn on> by <whoever> at <timestamp>"*.

Applications can employ this service to infer information about the environment or change their behavior depending on actions realized in the past. This service can also be useful for debugging purposes.

Rule Service

This service (García-Herranz, 2008) adds the possibility of storing rules in the blackboard to specify the behavior of the environment under some specific conditions. It is based on ECA Rules (Events-Conditions-Actions) that are composed by:

- **Events**. The reason why the rule should be triggered.
- **Conditions**. The context state that needs to be satisfied for detonating the action.
- **Actions**. The task that has to be accomplished.

This ECA rules can be used to express both reaction and transformation rules. The reaction rules are the ones that produce changes in the world as a reaction to a condition. The use of reaction rules allows modeling behaviors like user preferences. For example, a user can personalize the noise level of the environment. A rule can express that when the telephone rings if the user is in the environment and the TV is on, the TV should mute.

Transformation rules allow producing new information from other information. For example, a door is represented in a graphical interface by an icon that transforms depending on the state of the door. A rule can be used so that if it is opened, the

image will show an open door and if it is closed, the image will show a closed door.

Interface Service

The information stored in the blackboard can be also used by applications to provide interaction interfaces between users and the environment.

Since these environments provide new possibilities of interaction (Weiser, 1994), designers of the interfaces have to face new challenges (Shafer, et al., 2001). This interaction could be in different ways, oral interactions, gestures, tangible, etc. and also in many different devices, such as a PC, mobile devices (Eisenstein, J., et al. 2000), etc.

To allow the interaction, people who use the environment and the environment itself must share the same knowledge (Brujin, 2003). This common knowledge can be employed to obtain multiple and dissimilar interfaces.

Implementing an adaptive interface for each environment is an awkward task. This is solved by automatically generating the interfaces form the information stored in the blackboard. Since it is possible to monitor the changes of the environment, the interfaces reflect this transformation dynamically.

This service relates a description of the different ways of direct control to each element of the blackboard. Specifically, the direct controls of the interfaces are associated to the capabilities of the elements. This description of the interfaces is divided in three parts: one describing general properties of the representation, another one adding information related to the graphical interface and a third one that stores information for the oral interface.

- **General representation information**. This includes properties that are common for both oral and graphical interfaces.
- **Graphical representation information**. It describes the graphical interface: images, positions and sizes, interaction objects (button, slider bar, etc.)
- **Oral representation information**. Linguistic information is defined here, such as lexical and grammatical information.

We use this information to generate both graphical and oral interfaces (Montoro et al., 2006), although it could be used to deploy other kind of interfaces. As an illustrating example, the "hasStereoVolume" capability has two properties: volume level for the left speaker and the volume level for the right speaker. An audio source entity, or any other element with this capability, will have, among others, two interface descriptions, one for each property. From these two descriptions the graphical interfaces shows a slider in the audio source and the oral interface generates the dialogue to dim up and down the volume.

CONCLUSION

In this paper we have presented a middleware developed for ambient intelligence. Since developers have to deal with the problems that an intelligent environment presents, such as distributed and different nature technologies, hardware and software integration, different kinds of networks, new elements that appear in the system at any time, interface adaptation, etc.; a middle layer helps them to overcome them. Some of the characteristics of this system are:

- **From the data model point of view**. It separates functionality and data, letting applications to be spatially, temporally and also functionally uncoupled. The data-model remains unaltered, while applications may change. To develop adapted user interfaces becomes a straightforward task.
- **From the application model point of view**. The stored information follows a common data-model that acts as a logi-

cally central repository. It allows clients access the required information, regardless its nature.

- **From the communication model point of view**. There are two protocols to communicate with the blackboard. The first of them is a polling based protocol that allows retrieving information directly. The other one is based on a publish-subscribe mechanism, where the sources of information publish the changes and the consumers subscribe to those changes to receive the related information.

The system provides a set of operations to allow the communication with the clients. They do not have to worry about how the blackboard obtain or deduce any information or to know the internal structure of the whole system (shown in Figure 2).

We also consider that scalability is a critical characteristic of this middle layers. It has been guaranteed by the use of domains. A domain can exist by itself or can compound a bigger one, so the scalability is carried out automatically.

Finally, some services have been added to provide additional functionalities. Among them, a log service in charge of registering all the actions performed by the blackboard, a rule service to define rules that modify the behavior of the environment under some conditions and an interface service that adds information relative to the description of the user interfaces.

The work presented in this paper has been developed and tested in a real intelligent environment. We have adapted a laboratory furnishing as a living room and a workspace. It currently integrates a heterogeneous set of devices from different technologies (such as KNX, X10 or Phidget). Among them, we can find lights and switches, an electronic lock mechanism, speakers, microphones, a radio tuner, a TV set, RFID cards, etc.

Following the "Build what you use, use what you build", the laboratory is used in a daily basis by their members as a test bench for developed and new technologies. It allows direct control of its elements by means of manual control, a graphical user interface and a spoken dialogue interface and indirect control by means of a set of behavioral rules. We have also developed graphical user interfaces for mobile devices (iPhone and Android platforms), so a user can interact with the environment anywhere and at any time.

It also allows external collaborators and new members to develop new applications and interfaces that will be easily integrated in the environment. Among them we can find a gestural interface associated to a multi-touch table or new location and people recognition modules.

ACKNOWLEDGMENT

This work was partially funded by ASIES (Adapting Social & Intelligent Environments to Support people with special needs), Ministerio de Ciencia e Innovación – TIN2010-17344, e-Madrid (Investigación y desarrollo de tecnologías para el e-learning en la Comunidad de Madrid) S2009/TIC-1650 and Vesta (Ministerio de Industria, Turismo y Comercio, TSI-020100-2009-828) projects.

REFERENCES

Åkesson, K.-P., Bullock, A., Greenhalgh, C., Koleva, B., & Rodden, T. (2000). A toolkit for user re-configuration of ubiquitous domestic environments. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, Paris. ACM Press.

Ballesteros, F., Soriano, E., Leal, K., & Guardiola, G. (2006). Plan b: An operating system for ubiquitous computing environments. In *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications*, pages 126–135. IEEE Computer Society.

Brumitt, B., Meyers, B., Krumm, J., Kern, A., & Shafer, S. A. (2000). Easyliving: Technologies for intelligent environments. In *Handheld and Ubiquitous Computing, 2nd Intl. Symposium*, pages 12–27.

Choi, J., Shin, D., & Shin, D. (2006). Intelligent pervasive middleware based on biometrics. *Lecture Notes in Computer Science*, *4159*, 157. doi:10.1007/11833529_16

de Bruijn, J. (2003). Using ontologies. enabling knowledge sharing and reuse on the semantic web. Technical report, Technical Report DERI-2003-10-29, DERI, 2003.

de Ipiña, D. L., Vazquez, J., Garcia, D., Fernandez, J., García, I., Sainz, D., & Almeida, A. (2006). A middleware for the deployment of ambient intelligent spaces. *Lecture Notes in Computer Science*, *3864*, 239. doi:10.1007/11825890_12

Eisenstein, J., Vanderdonckt, J., & Puerta, A. (2000). Adapting to mobile contexts with user-interface modeling. In *wmcsa*, page 83. Published by the IEEE Computer Society.

Engelmore, R., & Morgan, T. (1988). *Blackboard systems*. MA: Addison-Wesley Reading.

Erman, L., Hayes-Roth, F., Lesser, V., & Reddy, D. (1980). The hearsay-ii speech-understanding system: Integrating knowledge to resolve uncertainty. [CSUR]. *ACM Computing Surveys*, *12*(2), 213–253. doi:10.1145/356810.356816

García-Herranz, M., Haya, P. A., Esquivel, A., Montoro, G., & Alamán, X. (2008). Easing the smart home: Semi-automatic adaptation in perceptive environments. *J. UCS*, *14*(9), 1529–1544.

Gelernter, D. (1985). Generative communication in linda. [TOPLAS]. *ACM Transactions on Programming Languages and Systems*, *7*(1), 80–112. doi:10.1145/2363.2433

Gómez, J., Montoro, G., & Haya, P. A. (2008). ifaces: Adaptative user interfaces for ambient intelligence. In *Proceedings of IADIS International Conference Interfaces and Human Computer Interaction*, pages 133 – 140.

Gong, L. (2001). A software architecture for open service gateways. *IEEE Internet Computing*, *5*(1), 64–70. doi:10.1109/4236.895144

Haya, P. A., Alamán, X., & Montoro, G. (2001). A comparative study of communication infrastructures for the implementation of ubiquitous computing. *UPGRADE. The European Journal for the Informatics Professional*, *2*, 5.

Haya, P. A., Montoro, G., & Alamán, X. (2004). A prototype of a context-based architecture for intelligent home environments. In *International Conference on Cooperative Information Systems (CoopIS 2004)*.

Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial Intelligence*, *26*(3), 251–321. doi:10.1016/0004-3702(85)90063-3

Kidd, C., Orr, R., Abowd, G., Atkeson, C., Essa, I., & MacIntyre, B. (1999). The aware home: A living laboratory for ubiquitous computing research. *Lecture Notes in Computer Science*, 191–198. doi:10.1007/10705432_17

Kindberg, T., & Fox, A. (2002). System software for ubiquitous computing. *Pervasive Computing, IEEE*, *1*(1), 70–81. doi:10.1109/MPRV.2002.993146

Le Gal, C., Martin, J., Lux, A., & Crowley, J. (2001). Smartoffice: Design of an intelligent environment. *IEEE Intelligent Systems*, 60–66. doi:10.1109/5254.941359

Montoro, G., Haya, P. A., Alamán, X., López-Cózar, R., & Callejas, Z. (2006). A proposal for an xml definition of a dynamic spoken interface for ambient intelligence. In Computer Science (LNCS), L. N., editor, *International Conference on Intelligent Computing (ICIC 06)*, volume 4114, pages 711–716.

Paternó, F., & Santoro, C. (2002). One model, many interfaces. In Kolski, C., & Vanderdonckt, J. (Eds.), *Computer-Aided Design of User Interfaces III* (pp. 143–154). Dordrecht: Hardbound. CADUI, Kluwer Academic Publishers.

Phillips, B. (1999). *Metaglue: A Programming Language for Multi-Agent Systems*. PhD thesis, MIT.

Ponnekanti, S. R., Lee, B., Fox, A., Hanrahan, P., & Winograd, T. (2001). Icrafter: A service framework for ubiquitous computing environments. *Lecture Notes in Computer Science*, *2201*, 56–77. doi:10.1007/3-540-45427-6_7

Rayner, M., Lewin, I., Gorrell, G., & Boye, J. (2001). Plug and play speech understanding. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue-Volume 16*, pages 1–10. Association for Computational Linguistics.

Reenskaug, T. (1979). Thing-model-view-editor, an example from a planningsystem. 12.

Román, M., Hess, C. K., Ranganathan, A., Madhavarapu, P., Borthakur, B., Viswanathan, P., et al. (2001). Gaiaos: An infrastructure for active spaces. Technical Report UIUCDCS-R-2001-2224 UILU-ENG-2001-1731, University of Illinois, Urbana-Champaign.

Shafer, S., Brumitt, B., and Cadiz, J. (2001). Interaction issues in context – aware intelligent environments. *Human – Computer Interaction*.

Tandler, P. (2004). The beach application model and software framework for synchronous collaboration in ubiquitous computing environment. [Special issue: Ubiquitous computing.]. *Journal of Systems and Software*, *69*(3), 267–296. doi:10.1016/S0164-1212(03)00055-4

Villanueva, F., Villa, D., Santofimia, M., Moya, F., & López, J. (2009). A framework for advanced home service design and management. *IEEE International Conference on Consumer Electronics, Las vegas, EEUU, January*, 26.

Wang, X. H., Dong, J. S., Chin, C., & Hettiarachchi, S. R. (2004). Semantic space: an infrastructure for smart spaces. *IEEE Pervasive Computing / IEEE Computer Society [and] IEEE Communications Society*, *3*(3), 32–39. doi:10.1109/MPRV.2004.1321026

Weiser, M. (1991). The computer for the twenty-first century. *Scientific American*, *265*(3), 94–104. doi:10.1038/scientificamerican0991-94

Weiser, M. (1994). The world is not a desktop. *Interaction*, *1*(1), 7–8. doi:10.1145/174800.174801