HI^3 Project: General Purpose Ambient Intelligence Architecture

A. Paz-Lopez and **G. Varela** and **J. Monroy** and **S. Vazquez-Rodriguez** and **R. J. Duro**[1]

Abstract. This paper presents a general purpose multiagent layered architecture in an AmI context. The main goal of this platform is to make the development, deployment and integration of high level AmI applications and services easy. It is known as the HI^3 architecture and it was designed following criteria such as modularity, scalability, determinism, fault tolerance, connectivity and security. In this paper we describe the architecture, its components and how they interact in a hypothetical scenario where we implement a test applications.

1 Introduction

In the last few years a new research area involving several disciplines has emerged. It has been called Ambient Intelligence (AmI), Intelligent Environments or Active Environments. Its relevance and interest arises from the changes it will introduce in the daily life of normal people.

AmI may be taken as a technological adaptation of the environment in order to facilitate daily tasks and make life more comfortable to its inhabitants. Through the use of computational and communications technologies and their integration into the objects that people use, the adaptation of the environment to the uses and habits of the inhabitants may be achieved. This is, obviously a change in the way things currently work, where it is the inhabitant or user of the environment the one that has to perform this adaptation.

Currently there are a lot of researchers working around these concepts, approaching them from multiple points of view. Broadly speaking, two approximations to address an AmI system may be identified. The first one of them is to create an ad-hoc solution to solve a particular problem. This may be appropriate whenever the solution does not need to be expanded to other environments. Obviously, here the limitations imposed are on reuse and the scalability. On the other hand, one could choose to create a general purpose AmI architecture that could be used to support any solution. It is on this second approach where the research presented here is focused. In particular we are working on a broader multidisciplinary project which we call HI^3 technology. Its main objective: to create Humanized, Intelligent, Interactive and Integrated environments.

In its basic nature, AmI systems depend on the appropriate integration and orchestration of multiple technologies and information sources in highly distributed and dynamic environments. This means that most AmI projects that address AmI from an integral perspective have required the development of middleware, to facilitate modularization, communications, integration and orchestration of the multiple elements that participate in an AmI system, as a major goal.

Some examples are projects like the MITs Oxygen Project, which has developed middleware components such as the Metaglue system [3] [7], a framework for intelligent agents, or the Pebbles and GOALS systems [8], that enable the development of small distributed software components and their coordination to carry out complex tasks. The iROS system [5] [4] was developed within the Interactive Workspaces project at Stanford University as a meta-operating system that facilitates the integration of diverse devices through an event based communications system and discovery and data interchange services. Some more recent projects may be cited, like those carried out within the Amigo Project [10] on the use of multiagent systems to achieve adaptability and reconfiguration in a semantic service based model by means of the dynamic composition of services. The developments within the CHIL (Computers in the Human Interaction Loop) project are of interest, especially those of the AIT (Athens Information Technology) [9] [6] on the development of an architecture for AmI systems that use abstractions to achieve technology integration and multiagent systems for the development of AmI services.

Its easy to see that most research related to the area pay special attention to the development of middleware frameworks that cover the division of tasks into distributed components, as well as the communications and interactions among them, facilitating the development of applications for AmI environments.

We are focused on the creation of generic software solutions to support the development of all types of applications in the AmI area with a special interest on providing transparent access to the broad spectrum of available hardware technologies. To achieve this, we have conceived a general purpose software architecture within the HI^3 technology. After a first analysis some basic requirements were established. The main idea behind our design is the necessity to manage multiple elements that fulfil different responsibilities in distinct levels of abstraction. This basic analysis continues by identifying other desirable requirements such as: Connectivity and communication abilities. To facilitate the cooperation of the multiple elements that will constitute a system. Modularity and reuse of components. To support the distribution of responsibilities across the system elements. Scalability. To adapt the system to the necessities of every environment by easily adding new elements. Security and information privacy. Because the system will be operating with personal and sensitive information about the people. Real time (to a certain extent). To provide the responsiveness that the users will expect of the system. Therefore, we have decided to conceive a multilayer design to meet the distinct levels requirement and to develop it through a multiagent approach to meet the multiple element requirement. This main design decisions also provide us with the tools to support other requirements such as modularity, thanks to the multilayer approach, and connectivity and scalability, through the use of multiagent platforms which are characterized by their distributed operation, proac-

[1] Universidade da Coruña, Spain, email: alpaz@udc.es

tivity and scalability.

Compared with the multiple existing developments in the area, we believe that this solution addresses their two most typical limitations:

- They are constrained to particular environments such as collaborative work, interactive workspaces, healthcare, etc. Examples of this is the iROS system, focused on workspaces, or the Amigo Project, focused on home environments.

- They adopt particular hardware solutions, obviating the problem of integrating multiple heterogeneous technologies, or limiting this integration to only a few of the most typical cases. Examples are the developments within the CHIL project, focused above all on audio/video devices, or the Oxygen project that is mainly focused on its own hardware solutions.

The paper is organized as follows: the HI^3 architecture and its components are described first. After this some design and implementation details will be presented. A multiagent technology approach was chosen for this purpose. Examples and applications are presented at the end of the paper, to show the application of the architecture to real cases.

2 HI^3 Architecture description

In order to provide support for the development and integration of applications and services in the framework of AmI, we have defined an architecture that complies with the HI^3 technology premises. In addition, this architecture, abstracts the access from services and applications to the physical devices, providing the capacity to transparently integrate multiple technologies present in a distributed AmI sensing and actuation environment.

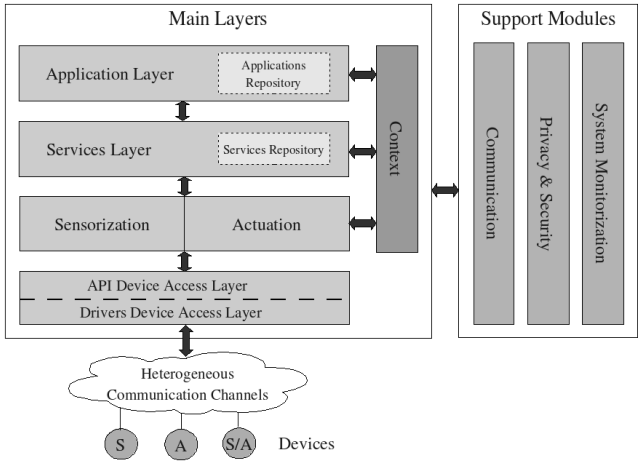

Figure 1. HI^3 architecture block diagram

Figure 1 displays the conceptual structure of the proposed solution. It is a layer based design that enables the division of system elements into levels, reducing the coupling between modules, facilitating abstraction and facilitating the distribution of responsibilities. Communication between layers is vertical so that a given layer can only communicate with the one right on top or below it. Within a layer, horizontal communication is allowed to permit its elements to cooperate to achieve greater functionalities.

- Device Access Layer (DA Layer): It is the layer that will contain those elements that are in charge of providing access to the physical devices. It is divided into to sublayers [11]:
  - Device drivers: It is a conceptual layer that contains the device drivers (both, proprietary and open) of the different devices connected to the system.
  - Device access API (Application Programming Interface): It defines an interface so as to homogeneously access devices and abstracts other components from the particular implementations and characteristics of the hardware devices. The elements of this layer play the role of adapters.
- Sensing and actuation layer (SA Layer): There are two clearly different types of components in this layer: sensing elements and actuation elements. Sensing elements are in charge of accessing the information provided by the sensor type devices (through the device access API). In the same way, the actuation elements must provide, again through the device access API, the necessary functionality to command the actuation of the actuator type devices. The elements of this layer are representations or models of the physical devices.
- Service layer: A service is defined as an element designed to solve a particular high level task and which, by itself, does not provide a complete solution for a system use case. Service composition is allowed, making possible for one service to use other services to carry out its tasks. The elements of this layer make use of sensing/actuation layer elements to access the devices. Services are managed through a module that provides a service repository. This repository enables the discovery of services needed by others to perform their task. Each service is defined following a template specified by the architecture. This template will force the formal definition of the input/output data managed by the service, as well as the specification of dependencies with other system elements.
- Application layer: This is the highest level layer and the one that hosts the elements representing and implementing components that solve particular functionalities a user expects from the system. These components make use of services registered in the system to carry out their tasks. Following the proposed layer based philosophy, applications will not access elements of the SA Layer directly. This task is delegated to the service layer. Application management is again achieved through an application repository.

Furthermore, there exists a common component for all the three higher level layers, the Context, its objective is the exchange and management of information generated throughout the system, and information necessary to characterize one entity situation, being an entity a person, an object or a place that is relevant to the interaction between a user and an application. One of its main goals is to represent the current state of the environment.

Finally, there is another group of architecture elements, the support modules. They include those components that solve basic tasks for the correct global operation of the system. They constitute a support for the proposed layer based architecture. Three main components stand out:

- Communications module: It is in charge of providing a transparent mechanism for synchronous and asynchronous communications between the components of the system. All of the elements will communicate using this module, ignoring their location, local or remote, as well as the type of hardware devices used.
- Security and privacy module: It manages all the aspects of the system related to authentication and identification, as well as access control to the different elements, data ciphering and privacy.

- System monitoring: It has two fundamental tasks:
  - Monitor the state of the different elements to detect anomalous situations, compile statistical data related to the operation of the system, etc.
  - Assure the correct operation of the system, relaunching elements when they are blocked or malfunctioning, managing multiple instances of some elements to achieve higher efficiency and scalability, etc.

This architecture represents a conceptual model that can be implemented using any software paradigm that supports the characteristics specified in its definition.

3 Design and implementation details

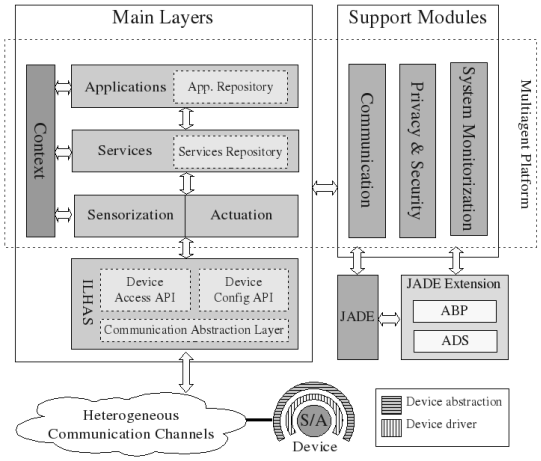

Figure 2. Implementation details block diagram

In order to implement the previously presented conceptual model, we have chosen a multiagent technology based approach. Many of the characteristics of this paradigm fit perfectly with the main objectives for our architecture, highlighting autonomous operation, distribution and collaboration on task resolution, proactivity, intrinsic communication capabilities, mobility and adaptivity.

Once a multiagent technology based approach was selected, we evaluated the possibility of using an existing agent framework versus the option of using an in-house implementation. Finally the JADE framework [2] was selected based on the following set of criteria:

- Widely extended use, especially in the research and university environment.
- Complies with the FIPA standard (Foundation for Intelligent Physical Agents) [1].
- Implemented in a multiplatform programming language like Java.
- Includes an implementation for mobile devices with limited capabilities.

Figure 2 displays a more detailed view of the HI^3 architecture. In it we observe how the main layers and support modules are realized within a multiagent platform. The main design guidelines are:

- The main elements of the high level layers are implemented as software agents that collaborate for tasks resolution.
- Support modules are integrated in the same multiagent platform. In some cases, like the communications module, they make intensive use of the facilities provided by the agent framework.
- We define an extension of the JADE framework to alleviate the coupling with the main elements of the architecture.
- The device access model is materialized in an abstraction of the devices that represent its functionality.
- We use ontologies to provide common semantics for agent collaboration.

The JADE extension component includes subsystems that provide common high level functionalities to the multiagent platform:

- ABP (Agent Behaviour Planner). Provides dynamic configuration capabilities for agent task execution plans.
- ADS (Agent Discovery Service). Component for registry and search agents within the system. The proposed implementation alleviates the JADE registry inefficiencies and provides a functionality based agent search engine using ontologies.

This multilayer, modular and scalable architecture together with the use of ontologies facilitates the existence of totally or partially redundant elements in the architecture. At the same time, it hides the complexity of their presence by implementing a search mechanism based on the functionality that these elements provide.

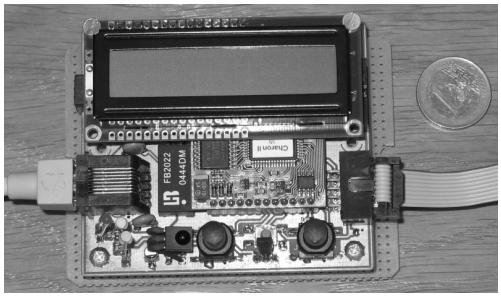

Figure 3. Hardware prototype of a sensing/actuation node

The device access model is based upon three basic abstractions:

- Unified and generic view of the different hardware devices available in an intelligent environment.
- Definition of a paradigm for the operating model of a domestic and social sensing/actuation network.
- Abstraction of the device network.

The first two abstractions provide access to the device based on its functionality. It is accessed through read/write operations over parameters that represent the information that a sensor/actuator is able to operate with.

The third one is achieved through a network layer and a node abstraction that enables the integration of multiple and heterogeneous device networks.

Within our AmI research we have designed and implemented a hardware device (see Figure 3) that would act as a low cost distributed sensing/actuation node that operates over open technologies and COTS (Commercial-off-the-shelf) hardware. We have also developed an ILHAS (Interoperability Layer for Home Automation System) [11] implementation for that device that transforms it into a good platform to integrate new sensors or effectors in an AmI environment. In this sense, this solution has been developed with the

main goal of representing an alternative to commercial buses and devices aimed at the home environment, characterized by their high cost and the use of proprietary technologies.

Currently we have developed the basic elements of the architecture that provide us with an basic platform to implement and deploy applications and services in the AmI area. This facilities will be illustrated in the next section with examples that were tested in our experimental environment.

4 Examples of applications

Next we will present some applications we have developed using the HI^3 architecture. These examples will serve to clarify some of the concepts presented and to illustrate how to take advantage of the different utilities provided by the platform to develop AmI applications.

We use our laboratory as an experimental environment, in it we have deployed multiple heterogeneous device technologies like a projector display, light controls, presence detectors or window/door opening sensors (all of them connected thought an EIB/KNX domotic bus), a PC with multichannel audio hardware, our own hardware node with a temperature sensor and switches, and finally, we have spread bluetooth beacons all over the laboratory to support a location service. Using this setup we have thought out and developed some examples that represent typical scenarios in an AmI environment like a users location tracking application. As a second application, we have considered managing a meeting room for presentations and work meetings.

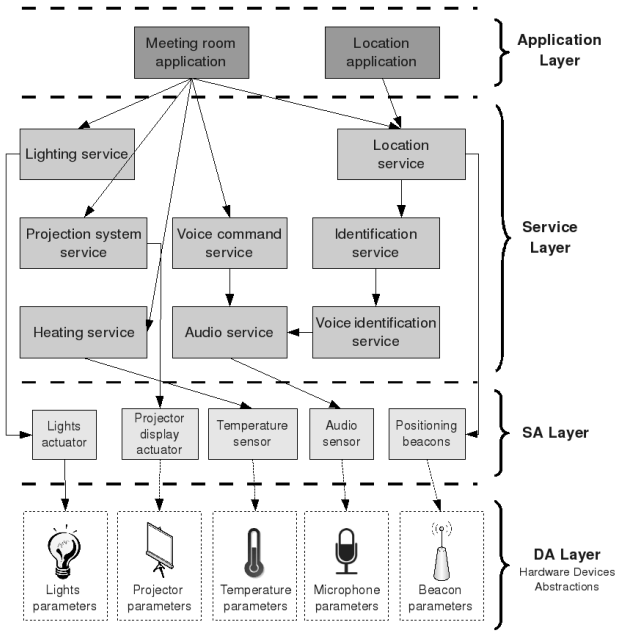

Figure 4. Component diagram of the proposed example

In the first case, we want to know where the users are located in the environment, so we want to have a location service that provides people's location information to other services and applications. A map application to graphically track the position of the people was built using this service.

In the second one, we want to control the physical devices that affect the configuration of the meeting room, such as light intensity, projector display or the temperature of the room, by giving the user a natural and adaptive interface to interact with the room.

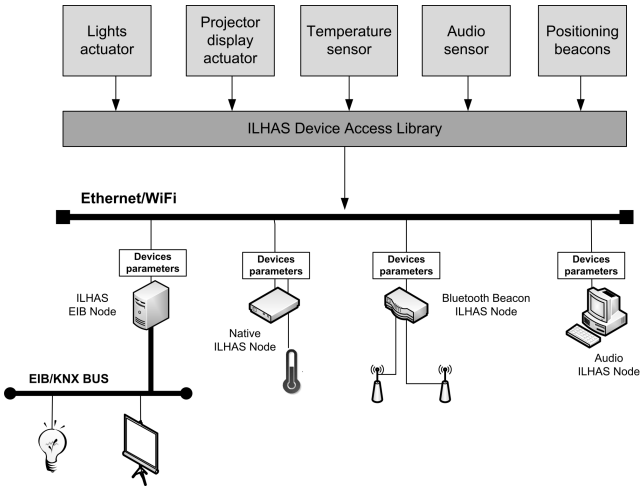

Figure 5. Abstraction of the hardware configuration for the examples

Figure 4 illustrates how the desired functionalities are divided in loosely coupled distributed components by following the architecture guidelines. To build the examples, it was necessary to develop multiple components on each one of the layers proposed; these components are agents (or even multiagent systems) that collaborate inside a multiagent platform. To implement those components, we made use of facilities such as the ABP that allows the dynamic composition and configuration of the component execution plan. It is achieved by using XML documents to declare the component's behaviour within a finite state machine. Thus making the development easy by allowing fast and easy changes to component behaviour. As will be seen, the Applications and Services Repositories have also been used through the ADS, which provides us with an ontology based component search engine that facilitates the decoupling of components, because one component only has to know what extra functionalities it needs to do its job, and the platform will automatically provide a component that fulfils these functionalities. Finally, we also make intensive use of the abstractions provided by the architecture, specially the hardware abstractions offered by the SA Layer and the ILHAS library. Those abstractions provide us with the ability to totally decouple our applications from the hardware idiosyncrasies.

For the first example proposed, we need an application that provides the graphical interface, in this case a simple map of the environment with the position of the people. This application is a software agent whose responsibilities are implemented through behaviours that are declaratively connected by using the cited ABP facilities. To do its job, the map application needs information about people and its location on the environment, so it makes use of the ADS to find another component that provides this functionality. The component that provides the location information will be a service component, because it doesn't provide functionality to end users, but to other system components. This is the Location Service (LS), which again is an agent that uses the ABP for its behaviour definition. The LS is a composite service that makes use of other services to do its job. Again it relies on the ADS to find its partners, in this case an Identification

Service that provides the identification facilities needed to identify the people tracked by the LS. The LS utilizes bluetooth beacons to triangulate people positions, the beacons are accessed through different sensor elements (again agents within the architecture found through the ADS) situated in the SA layer. Those elements act as proxies to the real hardware devices, using the device access library (ILHAS) to interact with them. It would be possible to use the ILHAS library from the services, but the architecture encourages the use of the SA layer, since it provides us with another level of decoupling, allowing the use of higher level abstractions of the devices (multiple device access, device redirection, redundant device access, etc.). Those SA layer elements are also accessible by their functionality through the ADS, and they use the ILHAS library to transparently (with regards to device technologies and locations) access the bluetooth beacons that are deployed on embedded PCs running our ILHAS Node Software.

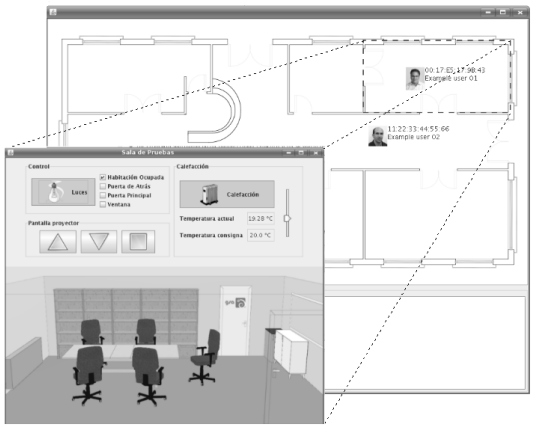

Figure 6. Example application screenshot

For the second example we want the system to autonomously manage the different configurations of the meeting room based on user preferences. We also want the user to be able to interact with the environment using his/her voice. To achieve this, as can be seen in Figure 4, we again make use of the architecture utilities, like the ABP and ADS, and reuse some of the services of the previous example, like the Location Service, which is required by the meeting room application to know in which meeting room (if there are multiple rooms) it has to apply the actions. In addition, new services are developed to process audio data, like a Voice Command Service, to detect selected commands in voice conversations (to power off the lights, start a presentation, etc.) and a Voice Identification Service, to identify the people involved in the meeting and adjust the environment to their preferences. These services use the ADS to find its partners as explained before, and use sensor elements (from the SA Layer) to access audio data, as well as sensors and actuators to control the domotic devices of the meeting room. Again, this SA element use the ILHAS library to homogeneously and transparently access the hardware devices, that are connected through multiple technologies.

These examples serve as an illustration of the facilities provided by the architecture for the rapid development and integration of new applications in the AmI area. These include service composition and reuse, decoupled collaboration between elements, dynamic addition or suppression of applications, services and sensors/actuators and homogeneous and distributed access to heterogeneous sensing networks.

In order to improve the architecture, service and application performances, we are cooperating with sectorial companies and professionals that can benefit from this technology. This has allowed us to test developments in real environments in addition to laboratory settings.

5 Conclusions

The architecture presented in this paper is a general purpose platform where we will be able to:

- Access hardware in a transparent and homogeneous manner.
- Easily develop high level AmI applications and services.
- Extend, modify and reconfigure an HI^3 system in order to adapt it to particular cases.

We have also developed a hardware distributed sensing/actuation node and an ILHAS implementation that transforms it into a platform to integrate new sensors or actuators in an AmI environment.

We are now focusing on the development of some modules of the architecture not considered in this paper, such as context, system monitoring and privacy and security modules. We are also working on applications and services supported by an HI^3 architecture in the scope of AmI and real installations such as buildings, hotels, retirement homes and offices.

ACKNOWLEDGEMENTS

This work was partially funded by the Ministerio de Educación y Ciencia of Spain through project DEP2006- 56158-C03-02 and Xunta de Galicia DXID under project PGIDIT06TIC025E.

REFERENCES

[1] FIPA Agent Communication Language Specifications - www.fipa.org.
[2] JADE. Java Agent DEvelopment Framework - jade.tilab.com.
[3] M.H. Coen, B. Phillips, N. Warshawsky, L. Weisman, S. Peters, and P. Finin., 'Meeting the computational needs of intelligent environments: The metaglue system', *Proceedings of MANSE'99*, (1999).
[4] B. Johanson and A. Fox, 'The event heap: a coordination infrastructure for interactive workspaces', *Mobile Computing Systems and Applications, 2002. Proceedings Fourth IEEE Workshop on*, 83–93, (2002).
[5] B. Johanson, A. Fox, and T. Winograd, 'The interactive workspaces project: experiences with ubiquitous computing rooms', *Pervasive Computing, IEEE*, **1**(2), 67–74, (Apr-Jun 2002).
[6] I. Pandis, J. Soldatos, A. Paar, and Reuter et al., 'An ontology-based framework for dynamic resource management in ubiquitous computing environments', *Embedded Software and Systems, 2005. 2nd International Conference on*, 8 pp.–, (16-18 Dec. 2005).
[7] B. Phillips, *Metaglue: A Programming Language for Multi-Agent Systems.*, Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1999.
[8] U. Saif, H. Pham, J. Mazzola Paluska, J. Waterman, C. Terman, and S. Ward., 'A case for goal-oriented programming semantics', *System Support for Ubiquitous Computing Workshop at the 5th Annual Conference on Ubiquitous Computing (UbiComp '03)*, (2003).
[9] J. Soldatos, N. Dimakis, K. Stamatis, and L. Polymenakos, 'A breadboard architecture for pervasive context-aware services in smart spaces: middleware components and prototype applications', *Personal Ubiquitous Comput.*, **11**(3), 193–212, (2007).
[10] M. Valle, F. Ramparany, and L. Vercouter, 'Dynamic service composition in ambient intelligence environments: a multi-agent approach', *Procediings of the First European Young Researcher Workshop on Service-Oriented Computing*, (2005).
[11] G. Varela, A. Paz-López, S. Vázquez-Rodriguez, and R. J. Duro, 'Hi3 project: Desing and implementation of the lower level layers', *Proceedings of the 2007 IEEE International Conference on Virtual Envirnments, Human-Computer Interfaces, and Mesarument Systems*, (2007).