A Middleware-based Approach for Context-aware Computing

Zigor Salvador-Artola, Mikel Larrea-Álava, Daniel Cascado-Caballero, José Luis Sevillano-Ramos, Roberto Casas-Nebra, and Álvaro Marco-Marco

Ubiquitous computing environments integrate a large number of heterogeneous devices, which convey an increasing level of complexity when developing ubiquitous applications. A solution to this problem resorts to the use of a software abstraction layer, known as middleware, which encapsulates the underlying elements of the environment and offers unified and standardised access to applications which need to make use of the resources of the environment. Moreover, a middleware layer can also provide high-level built-in services, such as context management services.

Keywords: Ambient Intelligence, Context, Heterogeneity, Middleware, Ubiquitous Computing.

1 Introduction

Pervasive applications which support smart environments integrate many electronic devices which are often very different in their nature, function, communication protocols and computational capabilities. Such heterogeneity comes at the price of increased levels of complexity during the development and maintenance stages of any modern pervasive application. In fact, developers have to deal with low-level hardware, communication networks and network protocols, orchestrating all of these different resources in order to provide high-level functionality to users

Authors

Zigor Salvador-Artola is currently a PhD candidate at the *Universidad del País Vasco* (UPV-EHU), Spain. He received his MS degree in computer science from the *Universidad del País Vasco* in 2003. He is currently a Research Assistant at the *Universidad del País Vasco*. His research interests include ubiquitous computing, interoperability and middleware-based architectures. He is the author of several papers on these subjects. <zigor@zigorsalvador.com>.

Mikel Larrea-Álava received his MS degree in computer science from the Swiss Federal Institute of Technology in 1995, and his PhD degree in computer science from the *Universidad del País Vasco* (UPV-EHU) in 2000. He is currently an Assistant Professor of Computer Science at the *Universidad del País Vasco*. His research interests include distributed systems, fault tolerance and ubiquitous computing. He is the author of several papers on these subjects. <mikel.larrea@ehu.es>.

Daniel Cascado-Caballero received his degree in Computer Science in 1996 and his PhD in 2003 (both from the *Universidad de Sevilla,* Spain). In 1992 he conducted research into 3D-location systems for the *Universidad de Castilla-La Mancha*. From 1997 to 1999 he worked at the Institute for Prospective Technological Studies (Joint Research Centre). In 1999-2000 he was a researcher supported by the *Universidad de Sevilla*, working on e-learning systems. Since October 2000 he is Assistant Professor in the *Universidad de Sevilla*. He is the author of several papers on wireless communications, mobile robots and rehabilitation systems. <danic@atc.us.es>.

José Luis Sevillano-Ramos received his degree in Physics (electronics) and his PhD from the *Universidad de Sevilla* (Spain) in 1989 and 1993 respectively. From 1989 to 1991 he was a researcher supported by the Spanish Science and Technology Commission (CICYT). After being Assistant Professor of Computer Architecture at the *Universidad de Sevilla*, since 1996 he is Associate Professor at the same University. He is the author of various papers and research reports on real-time communications and architectures, mobile robots and rehabilitation systems. <sevi@atc.us.es>.

Roberto Casas-Nebra received his MS degree in electrical engineering in 2000 and a PhD degree in electronic engineering in 2004, both from the *Universidad de Zaragoza*, Spain. Until 2007, he was assistant professor in the Univer*sidad Politécnica de Cataluña*'s Electronic Engineering Department and a member of the Technologies for Disability (Tecnodiscap) group at the *Universidad de Zaragoza*. Currently he is a researcher in Tecnodiscap group. His research interests include sensor networks, digital electronics, and assistive technology. <rcasas@unizar.es>.

Álvaro Marco-Marco received his MS degree in electrical engineering in 2000 from the *Universidad de Zaragoza*, where he is a PhD candidate in electronic engineering. Currently he is an assistant professor in the *Universidad de Zaragoza*'s Computer Science and Systems Engineering Department and a member of the Technologies for Disability (Tecnodiscap) group at the *Universidad de Zaragoza*. His research interests include sensor networks, digital electronics, and assistive technology. <amarco@unizar.es>.

and applications. The coordination of all the different factors proves quite challenging, as developers need to be aware of all the elements in the environment, all the way up from the low-level ones to the high-level ones, taking into account the behavior of those elements in all the architectural layers. Moreover, there are currently many different technologies which address the same functional requirements, making in-depth studies necessary in order to choose one technology over the other. Therefore, code optimization and debugging can arguably become a daunting task in such pervasive environments.

A solution to this particular problem can be found in middleware [1]. Middleware represents a software abstraction layer, which basically encapsulates the underlying elements of the environment and provides a unified way of making them accessible to high-level software applications. In addition to this unification, middleware can also provide additional services or functionality to applications which make use of it. For instance, and depending on the particular focus of the middleware, it can provide context management services, appliance control services, user management services, services related to security and others. Generally speaking, any system where seamless [2] integration of different devices and resources is a key to success can benefit from adopting a middleware-based solution. Pervasive computing environments, in particular, clearly benefit from using one or more standard abstraction layers, mainly due to the progressively growing number of devices that need to be integrated, and the logical degree of heterogeneity they convey.

The rest of the document is structured as follows. Section 2 deals with the state of the art in context-acquisition technologies and, in particular, technologies for location and sensor network technologies. Section 3 focuses in middleware as a tool for integration in ubiquitous environments. Finally, Section 4 analyzes some of the currently available middleware platforms and their trends.

2 Technologies for Ubiquitous Computing

A particularly relevant technological area related to ubiquitous computing is the one providing the tools for context information acquisition, processing, and its efficient usage by applications. Technologies designed to find out the physical location of people or objects, and technologies for the transmission and aggregation of generic information provided by one or more sensors are two clear examples of current and popular applications of ubiquitous computing.

2.1 Location Technologies

As Mark Weiser stated at the end of the 1980s, location is a pillar of ubiquitous computing. In contrast with outdoor location, where GPS and, more recently, GSM are the only available technologies, indoor location systems present a greater degree of heterogeneity, with a wide variety of technologies that allow for locating a mobile object or body: artificial vision, field sensors, radio-frequency, etc. As there is no standard indoor location technology, each solution is constrained to a specific situation. Moreover, these solutions have a relatively high cost for their features, which currently limits their generalization.

For outdoor location, well-known solutions exist, such as GPS [3], or the European version, Galileo [4]. Both are based on the classic multilateration idea, requiring the data from at least three satellites to compute a location, and usually a fourth satellite to validate that result. Location errors with these systems are within 5 meters in favorable environments, and it is possible to reduce them to 3 meters with DGPS which uses terrestrial stations. Another interesting point is that GPS is a widely adopted standard. On the other hand, its main drawback is that signal reception is not guaranteed indoors, or in cities with big buildings, which hinders its use in these scenarios [5].

GSM location also acquires a great significance due to the spectacular increase in mobile phone usage, and the FCC requirement of emergency call location traceability in the United States [6] and the European Union [7]. The easiest way to obtain the location of a mobile phone is placing the mobile terminal within the cell that it is connected to the Cell of Origin (COA), which can achieve an accuracy of several hundred meters in cities [5]. But this accuracy is not enough for some applications, and there are more sophisticated methods based on measuring the signal strength (RSSI), the angle of arrival (AOA) and particularly the time of arrival (TOA) or the time difference of arrival (TDOA). However, these techniques are highly affected by the interferences generated by the environment, commonly denoted as non-line-of-sight (NLOS) errors.

Focusing on indoor location, the number of existing typologies to perform location dramatically increases: ultrasounds, radiofrequency, artificial vision, magnetic field sensors, inertial sensors, etc. The systems that show a better coverage/accuracy/cost ratio are those which combine ultrasound technologies with radiofrequency, taking advantage of the different propagation speeds of both signals to precisely measure the time of flight. Each system uses different strategies, leading to differences in location accuracy —both spatial and temporal—, scalability, ease of deployment, adaptability to predefined profiles, centralized or decentralized management, etc.

AT&T Laboratories Cambridge developed Active Bat [8]. This system has a controller who sends a location query via radiofrequency. Then, the device to be located sends an ultrasound pulse at the same time that the controller sends a synchronization signal to an array of sensors placed on the building ceiling which are connected by wires. Each sensor measures the time between the synchronization signal and the ultrasound pulse arrival, and the measurements are processed by the controller which computes the location. In its last version [9], it is possible to locate three mobile devices within 3 cm up to 50 times per second. Its main drawbacks are the big infrastructure needed (a 1000 square meters area requires 750 sensors), and its dependence on the correct placement of sensors. Scalability, ease of deployment and the global cost are therefore the main drawbacks of this system.

The Cricket system [10], developed by MIT, uses beacons, distributed along the ceilings and walls of a building, which emit an ultrasound pulse at the same time as they send information via radiofrequency. Receivers placed on the device to be located listen to those signals and send the acquired information to an intelligent device connected to them. This device runs an algorithm that estimates the distances to the beacons and then, via triangulation, calculates the location of the mobile object. This system shows similar drawbacks to the ones present in the Active Bat system, requiring a network of approximately one sensor per square meter to achieve an accuracy of one meter. Another inconvenient is that, due to the high computational load, receivers must be connected to moderately intelligent devices, such as PDAs or laptops.

Finally, another noteworthy system developed by MIT is the one known as Constellation [11], which combines ultrasounds with infrared radiation instead of radiofrequency, and also uses inertial sensors to obtain a location by double integration of the acceleration. This system offers better features than most of the previously reviewed systems, with a high refresh rate and millimetric accuracy, providing even orientation information. It is robust and immune enough to occlusions, but requires a big infrastructure and has a limited coverage.

2.2 Sensor Networks

A sensor network is usually composed of a relatively high number of sensors (typically 100 to 1000) interconnected through wireless links. The topology of a sensor network is usually highly variable. In fact, the network autonomously organizes itself, with information routing, fault tolerance, and the like, performed in a cooperative way by the sensors. Sensors (or nodes) are usually low-cost and therefore have limited resources: battery, memory, computer power, bandwidth, etc. Particularly, energy consumption is a critical concern since it limits data transmission rate and range, as well as network autonomy. A broadcast communication paradigm is generally used, and traffic in these networks has some unique characteristics, such as being typically synchronous. For instance, in some sensor networks, all nodes activate themselves at the same time when they detect a change in one of the physical magnitudes being sensed (e.g. temperature).

Although significant advances have been made in wireless sensor networks (WSN), this is still an open research issue. Currently, the IEEE 802.15.4 is the *de facto* standard, but there exist a number of alternative or complementary technologies that we briefly discuss below.

Bluetooth [12] is a mature, widely accepted technology focused in wireless personal area networks, with data rates up to 1 Mbps and a range of 10 meters. Bluetooth devices form networks called *piconets* with up to 7 nodes (*slaves*) coordinated by a *master*. Although multiple piconets can be interconnected forming *scatternets*, the resulting number of nodes is still low for most typical sensor network applications. This fact, together with its relatively high power consumption, makes this technology an alternative for some specific situations, but not a reference technology for WSNs.

Ultra Wide Band (UWB) [13] is currently being introduced in the market, due to recent regulations on ultra-wide band communications in Europe and the U.S.A. This technology has been developed for high-speed personal area networks, with data rates of up to 500 Mbps across a distance of 2 meters, and 100 Mbps across 10 meters. However, the use of this technology is not only oriented towards high speed applications, as it is also being used in sensor networks with high bandwidth requirements (Wireless Multimedia Sensor Networks-WMSNs). The high data rate of UWB makes this the technology of choice for future wireless connections like WirelessUSB and WiMax. Furthermore, its adoption as a physical layer option for Bluetooth is being considered.

Wibree is a proposal from Nokia [14] for WSN where the Bluetooth physical layer is used and the higher layer protocols are simplified in order to guarantee lower power consumption, more adequate to the needs of WSNs (often built using battery powered devices). The proposal is very recent but it is rapidly gaining momentum and Wibree has just been included in the Bluetooth specification.

The Zigbee technology, based on the IEEE 802.15.4 standard, was developed by the ZigBee Alliance [15], and has become a *de facto* standard for low-power sensor networks. Currently, a high number of manufacturers offer hardware devices and development kits for this technology, such as the kits offered by Ember [16].

Among the most recent technologies, it is worth mentioning both ANT and SP100. ANT [17] is a technology for WSN nodes based on the wireless technology from Dynastream [18]. It offers a performance similar to that of Zigbee with slightly lower power consumption and a data rate close to that of Bluetooth. SP100, on the other hand, is a standard from Honeywell [19] for wireless communications in industrial and control applications, in direct competition with Zigbee and Bluetooth. It is expected that SP100 devices will have battery life of up to 10 years.

3 A Middleware-based Approach

The aforementioned example applications of location and sensor networks can be seen as specialised cases of ubiquitous computing. However, smart environments of the future will probably integrate such applications inside a wider collection of subsystems that will conform the pervasive environment and will therefore benefit from integrative middleware solutions in order to minimize the complexity of the system. We will consider three fundamental aspects of this type of generic smart environments: acquisition of contextual information, control of the environment and user interaction with the environment.

3.1 Acquisition of Contextual Information

In pervasive applications and environments, we define context as a representation of the knowledge that a system has about the state of the system itself. By definition, con-

text in pervasive environments is partial, due to constraints in sensing technologies and logical limits found in the algorithms which manage context. However, context information, although partial, is an invaluable resource for pervasive applications, as it allows applications to gather information about the real world, and adapt their behavior accordingly. Context is not just a snapshot [20] of a pervasive environment at a particular moment. It usually represents information from a given time-frame in the life of the pervasive environment, providing the basis for systems which learn from their past context, provide enhanced adaptation capabilities and therefore support proactive decision-making with varying degrees of autonomy.

Contextual resources include both the sensing hardware and the sensed information. Nowadays, many types of electronic devices, such as motes [21] and most handheld devices such as cell phones have sensing capabilities of one type or another. The most widespread basic sensors include temperature, humidity, acceleration and lighting sensors, while complex or applied sensors offer more refined sensing information, such as the readings provided by motion-detection, intrusion-detection or fall-detection sensors. The pool of raw sensor readings can be processed by an application to provide contextual information. Contextual information can be created using low-level context processing, such as evaluating individual readings, or it can be inferred and composed using high-level context processing techniques which take into account a part of the raw reading pool, or even the whole pool of readings [22].

Managing contextual resources poses various challenges to developers of pervasive applications: detection of existing contextual resources, data gathering from those resources, persistence of old contextual information, inference of high-level contextual information from raw readings, and others. A well-designed middleware module could take care of all those low-level aspects of context management, and provide applications with unified interfaces to access those resources in a simple and efficient way. The middleware could, in fact, provide additional functionality by partially processing contextual information and offering inferred context in addition to raw readings. Persistence and optimization of various management aspects of sensor networks and differences in the technologies used by those sensor networks would be transparent to the application developer using such a middleware module.

3.2 Control of the Envionment

Control represents the ability that pervasive environments and applications have to change the environment. These changes in the environment are achieved through controllable devices or appliances, such as TVs, VCRs, home automation kits, air conditioners, automated kitchens, security locks and many more, depending on the particular deployment environment. For instance, in a pervasive health-care environment users could control specific devices designed to augment their autonomy and comfort, such as intelligent wheelchairs, automated doors and windows, cooking appliances and such. Changes in the environment can be initiated by applications or users, and be reactive or proactive, depending on the particular situation or context of the environment when control is requested and always keeping in mind that proactive decision-making should respect the autonomy of users and be as non intrusive as possible. Commands are issued to devices using the most suitable communication media available in the environment, ranging from short-range wireless interfaces to long-range wireless interfaces, cabled computer networks and even power line communications [23].

Therefore, control resources include not only the devices or appliances which can be controlled, but also the communication media that makes control possible. Having many different communication technologies and device standards which are often not interoperable with one another, application developers have to decide beforehand which devices the application will control, and develop specific bridges to that particular technology or standard. This limitation translates into systems which only work in very specific locations and, even then, only when controlling very specific hardware. Thus, the variety of controllable resources is interesting in the theoretical level, but quite a hurdle when practical implementations of pervasive computing environments and applications are to be developed. Logically, as is the case with context and user interaction, decoupling applications and device-specific details by adding an abstraction layer on top of the device hardware level can help minimize the complexity of developing pervasive computing applications.

A middleware module for device and appliance control could take care of most of the complexity of discovering, setting up, and effectively managing all controllable resources in a given environment. The module could offer a

```
public interface ApplianceControlManager
{
   public ListOfAppliances getListOfAvailableAppliances (Environment selectedEnvironment);
   public ListOfCommands getListOfAvailableCommands (Appliance selectedAppliance);
   public void executeCommand (Command requestedCommand, Appliance selectedAppliance);
}
```

Figure 1: Example of a Programmatic Control Interface.

unified programmatic interface to applications, which would then have all the controllable resources of the environment at their disposal, regardless of the particular technology or group they belong to. An example programmatic interface is shown in Figure 1, to better illustrate the functionality offered by the middleware module. All together, such a module opens doors to applications which stay available even if devices change or get replaced by newer models or technologies. On top of abstracting the underlying network of appliances, the middleware module for control would have the added benefit of arbitrating control among applications, to avoid conflicting commands, set defaults, and issue permissions for exclusive control rights over certain resources. Overall, a middleware module for device and appliance control, combined with a suitable model for control and controllable resources can reduce the development time of pervasive applications which aim at changing the environment, be it autonomously or under the strict guidance of end users or administrators.

3.3 User Interaction

In the domain of pervasive computing applications and environments, user interaction represents the communication between a user and the environment. User interaction is mandatory for most pervasive applications, as they are built around users and their needs, although some particular autonomous computing systems could be in charge of an environment without the need for user interaction, and some very specific applications could also be autonomous in that sense. However, being an emerging technology, sending information to users and getting their feedback is still one of the main pillars of pervasive computing applications and environments alike.

Reduced to its essence, user interaction is a two-party communication paradigm. In spite of that, pervasive computing applications and environments transform this simple paradigm and introduce new players into the game. Due to the vast number of devices present in foreseeable pervasive environments, users will have a wide range of devices at their disposal to interact with the pervasive computing applications running in those environments. Conversely, applications running in pervasive computing environments will have control over many devices which could be used to perform user interaction in the most suitable way taking into account the context of the environment. Interaction renderers are devices which offer their resources to the system and the users, in order for them to communicate with each other. Due to the heterogeneity of devices (screen sizes, multimodal capability, computing power, etc.) and the similar heterogeneity of users (age, gender, cognitive capacity, mobility challenges, etc.), a proper model is required to describe the capabilities of interaction renderers and the users interacting with them.

Managing interaction in a pervasive computing environment can be a complex task due to the aforementioned heterogeneity of devices and types of users. In particular, an abstraction layer on top of interaction rendering devices could greatly benefit pervasive application developers, providing them with a unified pool of interaction devices which describe their capabilities in a standardized way and have well-defined programmatic interfaces which encapsulate their complexity. A middleware module could implement this abstraction layer, providing a service for applications which need to perform some sort of user interaction. Additionally, this service could offer other functionalities, such as automated setup of interaction devices, user authentication capabilities, multimodal renderers and others.

4 The Middleware Platform

A middleware platform is a software platform for the implementation of middleware architectures. The growing ubiquity of network connections, the Internet, and the proliferation of mobile devices and electronic devices of all kinds have proven the utility of the middleware concept and have, thus, lead the demand for commercial APIs and standards for interoperability and middleware development. Middleware platforms vary in size and form, but share a common objective which is to provide software developers with tools to implement the middleware solution they need in the smallest amount of time. General purpose middleware platforms can be divided in two big groups: open and closed middleware platforms. The former aim at interconnecting resources from different administrative scopes, requiring standardized markup languages such as XML and having performance limitations derived from being computing platform independent. The latter, on the other hand, aim at creating a closed middleware solution under the authority of a single administration, usually embracing a particular computing platform and being more efficient at the expense of interoperation flexibility. One of the clearest examples of open middleware platforms is found in the Web Services standard [24], while the closed middleware counterpart to that example is best characterized by the Jini middleware technology [25].

Middleware platforms are often service-oriented, and use modularity and distributed computing techniques in order to offer a service-oriented architecture where modules are interconnected with each other, offer services to the public and at the same time make use of services offered by other parties. Key services in service-oriented architectures for pervasive environments and applications include services to provide resource discovery, transactional support, substitution functionalities for thin clients, authentication services, and general administration services to remotely manage the life cycle of services in the pervasive environment.

Acknowledgments

This work has been carried out by the universities of the Basque Country, Seville and Zaragoza (*Universidad del País Vasco UPV-EHU, Universidad de Sevilla, and Universidad de Zaragoza*) as a part of the AmbienNet project (TIN2006-15617-C03), funded by the Spanish Government (MEC, *Ministerio de Educación y Ciencia*).

References

[1] P. Bernstein. Middleware: A Model for Distributed Services. Communications of the ACM, 39(2), pp. 86-97, February 1996.

[2] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. IEEE Personal Communication, pp. 10-17, August 2001.

[3] I. Getting. The Global Positioning System. IEEE Spectrum, vol. 30, no. 12, pp. 36-47, 1993.

[4] C. López-Bravo, M. Cano. Galileo: The European satellite radio navigation system. IEEE Global Communications Newsletter, 2002.

[5] S. Levijoki. Privacy vs Location Awareness. Seminar on Network Security, Helsinki University of Technology, 2003.

[6] FCC. Revision of the Commission Rule to ensure compatibility with enhanced 911 emergency calling system. Federal Communications Commission (FCC), Washington, DC, Technical Report RM-8143, 1996.

[7] EC. Directive 2002/58/EC on privacy and electronic communications. European Commission, Technical Report Directive 2002/58/EC, 2002.

[8] A. Ward, A. Jones, A. Hopper. A New Location Technique for the Active Office. IEEE Personal Communications, vol. 4, no. 5, pp. 42-47, 1997.

[9] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, A. Hopper. Implementing a Sentient Computing System. Computer, vol. 34, no. 8, pp. 50-56, August 2001.

[10] N. Priyantha, A. Chakraborty, H. Balakrishnan. The Cricket Location-Support System. International Conference On Mobile Computing and Networking, pp. 32-43, 2000.

[11] E. Foxlin et al. Constellation: A Wide-Range Wireless Motion-Tracking System for Augmented Reality and Virtual Set Applications. Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques, pp. 371-378, ACM Press, New York, 1998.

[12] Bluetooth web page. <http://www.bluetooth.com>.

[13] UWB web page. <http://www.uwbforum.org>.

[14] Wibree web page. <http://www.wibree.com>.

[15] The Zigbee Alliance. <http://www.zigbee.org>.

[16] Ember web page. <http://www.ember.com>.

[17] ANT web page. <http://www.thisisant.com>.

[18] Dynastream web page. <http://www.dynastream.com>.

[19] Honeywell web page: <http://hpsweb.honeywell.com>.

[20] J. Coutaz, J. Crowley, S. Dobson, D. Garlan. Context is key. Communications of the ACM, 48(3), March 2005.

[21] Sun SPOT project web page. <http://www.sunspotworld.com>.

[22] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Keränen, E. Malm. Managing Context Information in Mobile Devices. IEEE Pervasive Computing, vol. 2, no. 3, pp. 42-51, July-September 2003.

[23] PLC Forum web page. <http://www.plcforum.org>.

[24] Web Services Architecture: <http://www.w3.org/TR/ws-arch>.

[25] Jin web page: <http://www.jini.org>.

Abbreviations and Acronyms

ANT: a wireless personal network protocol.
AOA: Angle of Arrival.
API: Application Programming Interface.
AT&T: American Telephone & Telegraph Co.
COA: Cell of Origin.
DGPS: Differential Global Positioning System.
FCC: Federal Communications Commission (U.S.A.).
GPS: Global Positioning System.
GSM: Global System for Mobile communications (originally *Groupe Spécial Mobile*).
IEEE: Institute of Electrical and Electronic Engineers.
MIT: Massachusetts Institute of Technology.
NLOS: Non-Line of Sight.
PDA: Personal Digital Assistant (a handheld computer).
RSSI: Received Signal Strength Indication.
SP100: a standard for wireless communications.
TDOA: Time Difference of Arrival.
TOA: Time of Arrival.
UWB: Ultra Wide Band.
WMSN: Wireless Multimedia Sensor Networks.
WSN: Wireless Sensor Network.
XML: Extensible Markup Language.