The LAICA Project: Supporting Ambient Intelligence via Agents and Ad-Hoc Middleware

Giacomo Cabri, Luca Ferrari, Letizia Leonardi, Franco Zambonelli*
Dipartimento di Ingegneria dell'Informazione
**Dipartimento di Scienze e Metodi dell'Ingegneria*
Università di Modena e Reggio Emilia
{cabri.giacomo, ferrari.luca, leonardi.letizia, zambonelli.franco}@unimore.it

Abstract

Users' environments are going to be disseminated of intelligent devices and sensors that will coordinate each other to help users in their activities. This leads to what is called "Ambient Intelligence", i.e., environments that exhibits a certain degree of intelligence. Software agents are the natural candidates for this kind of task, thanks to their autonomy and the capability to act on behalf of their owners, resulting thus appropriate in the Ambient Intelligence area. In this position paper we discuss the use of agents in Ambient Intelligence, and sketch the relevant aspects of the LAICA project. In particular we focus on the design of a middleware for Ambient Intelligence that connects several kinds of agents and other distributed components.

Keywords: *Ambient Intelligence, Middleware, Agents, Collaboration*

1. Introduction

Ambient Intelligence (AmI) represents a new vision of everyday life, made up of communicating devices and sensors, which lead to a pervasive intelligence in the surrounding environment supporting the activities and interactions of the users. Ambient Intelligence aims at producing a significant change in the way people live.

Thanks to the growth of the digital technology, and the great availability of different kinds of devices, even portable, the human life has been more and more empowered by a "parallel" digital world. Nevertheless, the wide spread presence of electronic device does not create AmI, since these devices must coordinate and cooperate each other in order to support human activities.

To reach AmI, the user must perceive the environment she is living in as *intelligent*, even if the environment single parts are not intelligent at all. In fact, AmI does not imply that the environment (i.e., the environment components and or devices) must be endowed with a kind of artificial intelligence, but that it must be perceived as intelligent by the user, for example reacting and anticipating her actions on the basis of past events or her preferences.

In other words, AmI proposes to enhance the quality of life, offering to users relevant services anywhere and at anytime, placing the users themselves at the centre of the system, while the latter remains at the edge.

Thanks to the exploitation of AmI, human activities can be supported by a digital environment, reducing time wasting and leading to a simpler life in every-day actions, or even improving results in enterprise activities. In fact, AmI is not tied to a specific human activity or environment, but can be applied to different scenarios and situations. Thus for example we can find AmI systems for the management of hospitals and healthcare in general [RodFPV], and in enterprises as well [RivVDA05], where AmI can both support worker activities and allow enterprises to be a part of several networks at the same time without requiring deep changes. Furthermore, since AmI proposes to improve and ease the participation of individuals in the society, it can be applied also to an e-government scenario [RivVDA05].

Starting from the above consideration, it can be stated that AmI proposes to make services available and user-friendly in order to ease and support human interactions., Thanks to the today wide-spread presence of devices that leads to ubiquitous computing scenarios, and since AmI requires to be distributed in the environment, AmI can be built over ubiquitous computing, coupling the two paradigms.

In this scenario, software agents represent one of the most promising technology to deal with AmI, since they can be successfully exploited in ubiquitous computing too, thanks to their capabilities of being autonomous, reactive, proactive and social [CabLZ00, Jen01, CabFL05]. Also mobility is an interesting feature, which can be fruitfully exploited in AmI.

In this paper we explain the motivations of exploiting agents in Ambient Intelligence, and introduce the LAICA

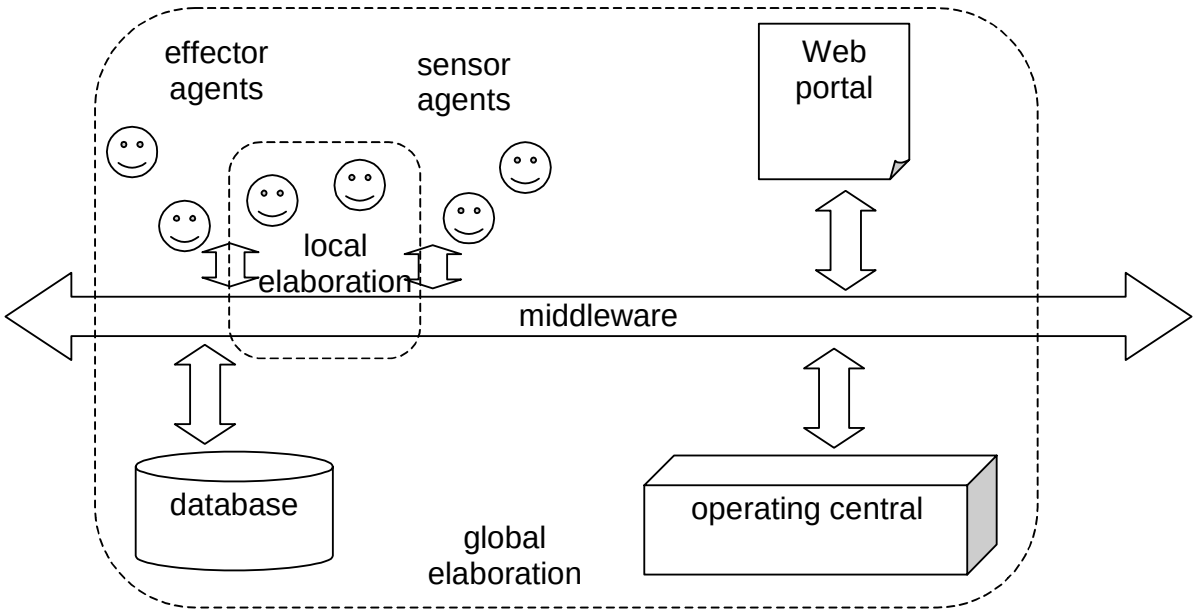

Figure 1. The LAICA architecture

project, in which we are involved with regard to agent interaction management.

The paper is organized as follows. Section 2 discusses the use of agents in Ambient Intelligence. Section 3 presents the LAICA project. Finally, Section 4 reports conclusions.

2. Ambient Intelligence and Agents

Agents, thanks to their capability of both executing in a proactive way and reacting to environment changes, can naturally deal with dynamism, heterogeneity and unpredictability [Jen01]. Moreover, the capability of dealing with unexpected situations as part of their intrinsic behavior, rather than in terms of "exceptions", enforces their adaptability to dynamic scenarios. Finally, their sociality leads to autonomy in interactions, allowing scalable decompositions of applications in terms of a decentralized multi-agent organizations [DemR96], and enabling interaction among agents not only belonging to the same application, but also to different ones, as happens in the people real world.

In particular, agents have been exploited to implement Ambient Intelligence in indoor environments, such as intelligent houses [Hag04], and even in healthcare system [RodFPV]. The key idea is that AmI needs a kind of ubiquitous computing in order to model and manage a whole environment. Since software agents represent a good paradigm to model an ubiquitous computing based environment [CaiLR02], it should be clear why several AmI approaches, including the one described in this paper, exploits agents. Thanks to the use of several agents, the whole environment can be modeled as a group of distributed services, which could be requested from users in different situations. Furthermore, the request for a service may produce the request to other services, leading to the need for cooperation and coordination among the environment agents.

Coordination is an important issue in multi-agent systems [Bus03], and several approaches have been proposed [CabLZ00]. We face this issue inside the LAICA project explained in the following.

3. The LAICA Project

LAICA stands for *Laboratorio di Ambient Intelligence per una Città Amica* (in English means: *Laboratory of Ambient Intelligence for a Friendly City*) [LAICA], and it is a regional project that involves universities, industries and public administrations. The main aim of the LAICA project is to define innovative models and technologies for Ambient Intelligence in an urban context, with particular regard to security issues, enabling AmI even at an enterprise level. To this purpose, LAICA propose an architecture with different components (see Figure 1).

The *sensor agents* are agents in charge of managing information acquired by devices. For instance, we make use of video cameras to acquire information about the traffic in the streets of the city. Modern cameras have embedded programmable intelligence so that they can be considered as agents. Old cameras can be attached to computational units where an agent executes, controlling the acquired information.

The *effector agents* are agents that can perform actions

on the ambient where they live. For instance, such agents can control traffic lights, alarms, and so on.

All interesting information data is recorded in a *database*, which is in charge of keeping track of whatever happens.

The *Web portal* is an important component of the LAICA architecture. From the one hand, it allows to control the whole system by means of a technology that is well-established, wide spread, user friendly, and usually allowed by firewalls. On the other hand, it enables the publishing of different kinds of information about the city and the current situations.

Finally, the *operating central* is the core of the architecture, in charge of instrumenting all other components in a fruitful way. Note that, even if this is a centralized component, the intelligence of the whole ambient infrastructure is spread among the agents and the middleware.

The middleware is the only component of Figure 1 not yet explained. First of all, it must be noted that the computing facilities (i.e., the hosts where the *sensor* and *effector* agents executes) are distributed and accessible in several mode on demand and as needed. Furthermore, computing facilities are heterogeneous, thus it is quite complicated to make interactions happening between them. Finally, all provided services should be both accessible from high power terminals (e.g., a workstation) and low power terminals (e.g., a PDA), in order to really support human activities without constraints. To face the above problems, LAICA proposes to manage and drive coordination of data and services through a middleware, which (i) it can know the location of data and services requested, (ii) it can translate data and commands between heterogeneous devices and agents, in order to make the coordination effective and (iii) it is in charge of performing those computations that the devices cannot do due to their low performances.

Inside the LAICA project, we are involved in the definition of the above middleware, which will be detailed in the following subsections.

3.1. Middleware Requirements

As already stated in the previous subsection, the middleware represents the "logic glue" among the different components of the LAICA architecture.

The ambient that LAICA aims to cover is quite wide, since it concerns a whole city. This implies a network infrastructure (wired and wireless) that is pervasive in the city and allows all the components to interact. Nevertheless, that interaction between components cannot be direct, since this would make interactions very complex and inefficient. Complex because the interaction logic should be embedded in every single service agent (either a sensor or an effector one); inefficient since it will be possibly unstructured, uncontrolled and the data exchange could imply the transmission of useless data

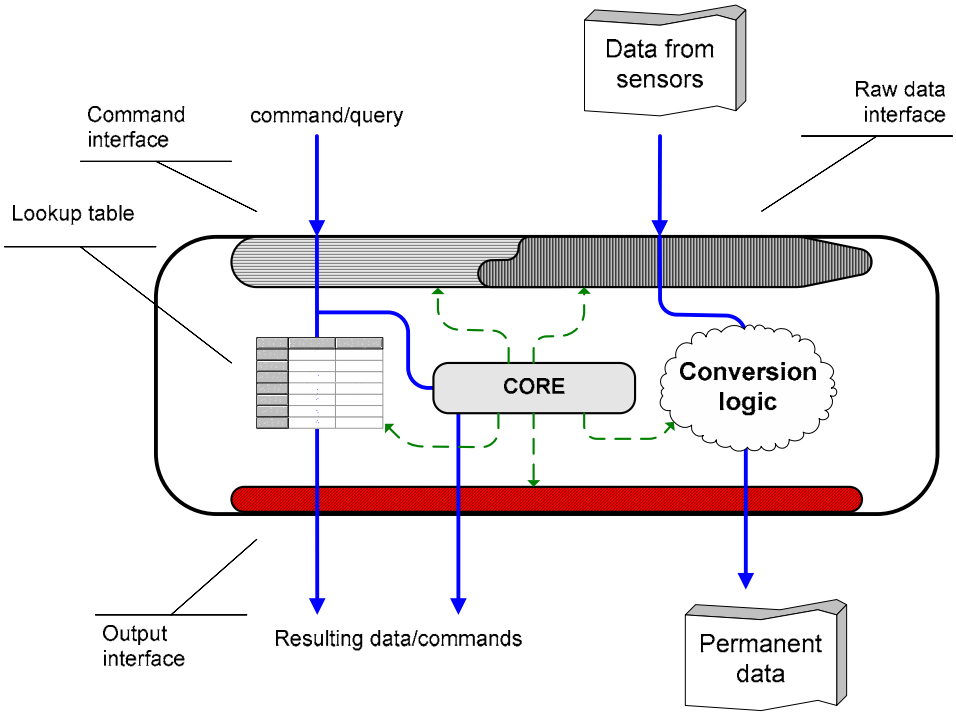

Figure 2. The LAICA middleware

along with useful data. For the above reasons, the middleware is in charge of managing and allowing interactions among LAICA components. This does not mean that every interaction must happen always through the middleware itself, because this could lead to performance problems. Instead, the middleware is in charge of making the interaction possible, for example providing the location of the counterpart of a specific service, or translating data and command when needed. For example, let us suppose that a video camera must coordinate with another video camera to make possible the tracking of a moving entity. In this situation, the first camera could simply ask the middleware where the camera nearer to the scene to track is, and once the middleware has answered, the two cameras could interact directly. A different situation could happen when a position sensor needs to activate a camera: since the involved effector agents comes from different devices, they probably are unable to coordinate themselves with a common protocol, thus the middleware must manage the whole interaction. Thus the LAICA middleware must manage interactions in a smart way, allowing them directly if possible, and driving them when needed, but always keeping track of what is happening on the whole system. This latter feature, that is having a panoramic view on the ongoing interactions, allows the middleware to play as the intelligent part of the LAICA architecture and, as a consequence, to make the user perceiving the whole system as intelligent.

Incoming data could require a transformation not only because it must be exchanged between two different components of the LAICA architecture, but also because it must be stored in the database (see Figure 1). Since LAICA aims at working with a lot of sensors, that means a lot of incoming data at the same time, it is clear that it is impossible, and furthermore it does not make sense, to store all the incoming data. For example, storing several hours of night images where nothing happens is not useful for any kind of study. For this reason, incoming data that must be stored can be converted by the middleware in a smart way, in order to focus and store only useful information (e.g., an image where a man is walking).

With regard to interaction, it is important to distinguish between *local collaboration* and *global collaboration.* The former happens among a few agents running in the system, without affecting the whole environment. This is the previous example case, where a camera coordinates with another camera to track a scene or a moving entity. Since local collaboration involves only a few agents, it implies also a *local elaboration*, which means that data is mainly locally elaborated by the corresponding effector agents.

Global collaboration, instead, affects the whole system. For instance, a sensor can report an overwhelmed condition in a street, that will activate a traffic light effector, then a camera effector to monitor the situation in the street, as other traffic lights as well in order to suggest different routes, and so on. In other words, in this case, the whole system is affected and must re-arrange its status to cope with the new situation. Similarly to local collaboration, global collaboration implies a *global elaboration*, since data must be evaluated by the whole system (i.e., every running agent) in order to react and keep the system under control.

3.2. Middleware Definition

The LAICA middleware is thought not only as a connection infrastructure, but also as a service provider.

To this purpose, the middleware can accept two kinds of incoming data: *raw data* and *commands*. The raw data represent information coming from the sensor agents and should be recorded in the database after a possible translation or elaboration; commands are issued by the components to obtain a service, for example to activate a sensor.

Since the raw data flow can be very rich, and it must be evaluated rapidly, to avoid the system collapse, we have provided the middleware of a double input interface (see Figure 2), one of which is dedicated exclusively to the incoming raw data. In this way, the middleware will not receive commands and raw data mixed up, and thus will know how to treat data coming from each interface.

Raw data will be immediately passed to the *conversion logic*, that will decide if the data must be stored or not, and in the case if it can be stored as it is or it must be translated. Before performing a translation, the conversion logic component must perform an analysis phase, in order to establish if the data can be thrown seamlessly or not, if it can be compressed, etc.

Commands can be passed to the *core* or to the *lookup table*. The lookup table is a repository that contains information about all other LAICA components. In other words, the lookup table knows where a service or specific information can be found, who (i.e., which agent) owns it, and can keep track of other information, like the protocol to adopt in order to cooperate with a specific agent.

The core represents the “brain” of the middleware, and manages all the middleware activities, such as translations, inputs and outputs (commands from the core to other middleware components are represented with dash-arrows in Figure 2). Nevertheless, the core is a kind of lazy component, since it instruments all other middleware components (such as the conversion logic or the lookup table) when needed. For example, the core can order to the conversion logic to increment the throughput (i.e., to reduce the pre-translation analysis time) because the middleware is becoming overloaded by raw data. Similarly, it can order a lookup table refresh since a lot of entries have not been queried since a while. Finally, the core can drive the input and output interfaces of the

middleware in order to isolate the middleware itself from one (or more) effector agents that are not working well (e.g., producing a lot of data).

Starting from the above considerations, it should be clear that the core plays as a "supervisor" of the middleware activities, but does not perform directly them. For example, the core does not see or evaluates the data flow, since the two input interfaces are designed to directly pass the data to the appropriate middleware component without querying the core. This is due in particular for performances and to keep the middleware as robust as possible. In fact, building the middleware itself as a set of cooperating components allows us to keep a part of the middleware working even if another part is no more. For example, a crash in the core will allow the middleware to work (even if not at its best) as conversion logic, without losing the raw data flow.

Both types of incoming data (raw data and commands) are coded using XML, but with two different schemas, since raw data have different properties from commands. The use of XML allows us to keep a high degree of interoperability among different entities and agents, and to evaluate the data in a short time. Since data coming from sensors and effector agents is of a multimedia kind (i.e., images, movies, audios, etc.), we need an XML schema to deal with this kind of data. At the moment, the MPEG-7 [MPEG-7] standard seems to be the better one, but we are still evaluating which schema could be the best one for the LAICA system.

The middleware presents an output interface, as shown in Figure 2, that is used to reports results of a command evaluation to other LAICA components or to output the data from the raw data stream. Unlike the input case, which makes available two separated interfaces, the output case has only one interface. In fact, it does not matter what kind of output the middleware is producing (commands or data), since it is always used as data for another component (e.g., the database or an effector agent). Thanks to its output, the middleware can coordinate all the component activities, making autonomous and intelligent decisions. It is important to note that the middleware contains only "coordination logic" because the logic of the management of the ambient is left to the operating central, and the recording of the information is in charge of the database.

3.3. Being an AmI Middleware

To be perceived as intelligent, the middleware should not only react to incoming commands, but must drive the whole system during its life. This means that the middleware must be able to recognize a global collaboration (which implies global elaboration) from a local one (which implies local elaboration).

It is important to note that the difficult in recognizing the kind of collaboration only happens when the interaction is a consequence of an external command. In fact, if the middleware does issue a command, it knows what the impact of the command will be, and thus which kind of collaboration will arise. If the command comes from the outside, instead, the middleware could not know what the command will produce until it is executing. For example, if an effector agent placed near a camera orders to the middleware to "activate the nearer position sensor", the middleware can consider this collaboration as local, since it involves only a few agents. However, if the command is "activate the nearer traffic light", this command can lead to a global collaboration, since managing one traffic light will affect all the others, and maybe could require the activation of cameras to monitor the traffic situation and so on.

To cope with the different kind of collaboration, the middleware can exploit the lookup table, to keep track of dependencies between agents running in LAICA. Thanks to this "dependency list", embedded in the lookup table, the middleware can analyze an incoming command, understanding which agents will (un)directly affect its execution, thus activating a kind of proactive reaction to quickly adapt the system. Of course, the dependency list cannot be statically filled, since dependencies can change at run-time, thus the middleware must be able to change the content of the table in order to better predict the collaboration that arises from a specific command. Furthermore, similarly to what happens for agent roles in [CabFL05], commands can be issued by a *command descriptor*, that allows the middleware to better understand the impact of such command on the whole system. We are still evaluating this idea, that requires that commands come along with a semantic description of themselves, thus the middleware can try to understand and predict what their execution will imply.

Another detail about the middleware is that it can exploit mobile agents to face unordinary situations. For example, if an agent (either sensor or effector) is no more responding to the middleware commands, the middleware can send a mobile agent to the device the former is attached to, in order to evaluate and try to locally correct the situation. Another example comes from an overloaded situation: if the middleware is overloaded by raw data that must be translated, it can send mobile agents to devices in order to perform locally the translation (if the device has enough computational power), thus the raw data must then just flow through the middleware itself. Moreover, the middleware can manage changes or updates sending to devices new versions of running agents.

4. Conclusions

In this paper we have discussed the exploitation of software agents in Ambient Intelligence, and has pointed out the importance of coordinating them.

The LAICA project aims at providing Ambient Intelligence at urban level, and relies on sensor and effector agents. In this project, our task is to support the interaction and the coordination of agents and of architecture components in general. To this purpose, we propose a middleware that enables interactions between components in a smart way. We have sketched the design of the middleware and the aim of the different internal components.

We are going to implement the middleware using the J2EE technology, which provides interesting features for our aim, since it is service-oriented and multi-platform, and offers support for wide-spread applications in enterprise environments.

Acknoledgments

Work supported by the Italian MIUR and CNR within the project "IS-MANET, Infrastructures for Mobile ad-hoc Networks", by the MIUR within the project "Trust and law in the Information Society. Fostering and protecting trust in the market, in the institutions and in the technological infrastructure", and by the project L.A.I.C.A. (Laboratorio di Ambient Intelligence per una Città Amica), funded by the Regione Emilia-Romagna, Italy, under the initiative 1.1 "Programma per la ricerca su prodotti e servizi innovativi" of the development project "Piano telematico regionale".

References

[Bus03] P. Busetta, M. Merzi, S. Rossi, and F. Legras, "Intra-Role Coordination Using Group Communication: A Preliminary Report", Lecture Notes in Artificial Intelligence n. 2922, Stringer Verlag, 2003.

[CabFL05] G. Cabri, L. Ferrari, L. Leonardi, "Injecting Roles in Java Agents Through Run-Time Bytecode Manipulation", IBM Systems Journal, February 2005, Vol. 44 N.1, pp.185-208

[CabLZ00] G. Cabri, L. Leonardi, F. Zambonelli, "Mobile-Agent Coordination Models for Internet Applications", IEEE Computer Magazine, Vol. 33, No. 2, pp. 82-89, February 2000.

[CaiLR02] G. Caire, N. Lhuillier, G. Rimassa, "A communication protocol for agents on handheld devices", Workshop on Ubiquitous Agents on embedded, wearable, and mobile devices, July 2002, Bologna, Italy

[DemR96] Y. Demazeau, A.C. Rocha Costa, "Populations and Organizations in Open Multi-Agent Systems", the 1st National Symposium on Parallel and Distributed Artificial Intelligence, 1996.

[Hag04] Hagras, H.; Callaghan, V.; Colley, M.; Clarke, G.; Pounds-Cornish, A.; Duman, H.; "Creating an Ambient-Intelligence Environment Using Embedded Agents", IEEE Intelligent Systems, Volume: 19, Issue: 6, Nov.-Dec. 2004, Pages:12 – 20

[Jen01] N. R. Jennings, "An agent-based approach for building complex software systems", Comm. of the ACM, Vol. 44, No. 4, pp. 35-41, 2001.

[LAICA] The Laica Official Web Site: http://www.laica.re.it

[MPEG-7] "MPEG-7 Overview", available at the MPEG official web site: http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm

[RivVDA05] G. Riva, F. Vatalaro, F. Davide, M. Alcaniz, "Ambient Intelligence: from Vision to Reality", Chapter of the book "Ambient Intelligence", available online at http://www.vepsy.com/communication/volume6.html

[RodFPV] Rodriguez M., Favela J., Preciado A., Vizcaino A., "An Agent Middleware for Supporting Ambient Intelligence in Healthcare", Workshop on Agents Applied in Health Care, August 2004, Valencia, Spain