Agents and Ambient Intelligence: the LAICA Experience

Luca Ferrari, Giacomo Cabri*
Dipartimento di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia
Via Vignolese, 905 – 41100 Modena – ITALY
{ferrari.luca, cabri.giacomo}@unimore.it

Franco Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria
Università di Modena e Reggio Emilia
Via Allegri 13 - 42100 Reggio Emilia – ITALY
zambonelli.franco@unimore.it

Abstract

Ambient intelligence generally refers to scenarios of computationally enriched environments enabling us both to better interact with the physical world and to integrate in the physical world smart functionalities. In this context, multiagent systems are a natural paradigm to develop and deploy dynamic and situation-aware ambient intelligence services. Here we present and discuss our experience in the development of agent-based ambient intelligent services for the city of Reggio Emilia, as performed in the context of the LAICA project.

1 Introduction

The general vision of "ambient intelligence" (AmI) considers the possibility of enriching our everyday environments (from cities to offices and houses) with sensing, computing, and communication capabilities. This can be used to provide a variety of "ambient" services enabling us to increase our effectiveness in understanding and interacting with the physical world [Hagras et al., 2004, Riva et al.]. Such services may span from simply providing some sensed environmental information (e.g., in the case of an urban scenario, signaling the presence of unusual overcrowdings in some parts of the city) to more complex services able to orchestrate the distributed components of the ambient infrastructure (e.g., a service to adaptively brand environmental cameras based on specific situations to be monitored).

Technological advances are making the AmI vision more and more real [Estrin et al., 2002], however, the question of how to effectively develop and deploy ambient services, able to get the best from the ambient infrastructure and serve users in a flexible, adaptive, and situation-aware way, is still open.

In this paper, we argue that multiagent systems [Jennings, 2001] represent the most suitable paradigm for the development of AmI services (Section 2). In fact, the concept of agents as adaptive autonomous entities that flexibly interact with each other and are situated in some sort of "environment" perfectly maps into a scenario of ambient services, in which a variety of typically distributed software components have to coordinate with each other to provide services on the basis of the current environment situation.

Starting form these basic considerations, the paper then focuses (Section 3) on our experience with the development of an agent-based infrastructure for ambient services in an urban scenario. Such an activity has been performed within the LAICA project [LAICA] (Italian acronym for "Ambient Intelligence Lab for a Friendly City"), funded by the Regione Emilia Romagna and aimed at enriching with cameras and sensors selected zones of the city of Reggio Emilia, so as to provide monitoring and alert services to both urban police and citizens.

Two selected case studies, describing two specific services that have been developed in that context, will be presented to clarify our approach and to better ground our general considerations (Section 4).

A discussion of related work in the area (Section 5) and some final remarks (Section 6) conclude.

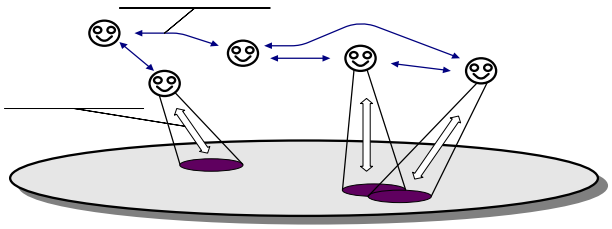

Figure 1. The general characterization of multiagent systems.

2 Agent-Based Computing and Ambient Intelligence

According to the more general characterization, agents are autonomous software components, able to sense and to affects their operational context, and typically flexibly interacting with each other to orchestrate the achievement of goals beyond their individual capabilities [Jennings, 2001; Zambonelli, 2003]. From the

* Work supported by the Regione Emilia Romagna within the LAICA (Laboratorio di Ambient Intelligence per una Città Amica) project.

software engineering viewpoint, this leads to the architectural characterization of Figure 1.

Multiagent systems researchers often claim the general-purpose nature of multiagent architectures and the advantages of applying it to a variety of scenarios. In AmI scenarios, agents can be exploited to represents components and humans, and the multiagent system paradigm can be adopted to implement the whole environment (see Figure 2).

In an AmI environment, some kind of hardware infrastructure must be provided, typically consisting of a variety of computer-based devices, sensors, actuators. These can either capture properties of the physical world or turn them into some accessible digital format, or can transform digital commands into specific physical actions. Thus, the AmI infrastructure perfectly maps into an abstraction of a multiagent system environment, acting as the medium via which interactions with the physical world can be enforced.

To provide services at the user level, as well as to control and coordinate activities over the physical world, some kinds of software service components must be provided to include the necessary logics. In general, service components of this kind should be able to properly interface with one or more devices of the AmI infrastructure and – to avoid centralization points and increase reliability – should be made capable of autonomously performing the necessary activities interface with such devices and to control them with appropriate actions. In other words, being situated in that part of the physical environment associated to that devices, and being autonomous in their actions, such service components should be considered (by explicit choice or *de facto*) as agents.

Please note that, in general terms, one can consider that the same sensor/device can be accessed by several agents and, on the other hand, the same agent can access several devices. This allows agents and sensors to spread information and to coordinate with each other via the mediation of the environment (i.e., of the AmI infrastructure) in order to become aware of what it is happening around. Consider for instance a camera that issues data related to a person that is running at nighttime. An agent catches such information and wakes up an audio sensor it controls. Another agent can then start "listening" to that audio sensor and possibly recognize if the person is screaming or laughing.

In general, as in the above example, information is scattered across the environment, and several agents can be in need to be involved to really understand what's happening. Thus, most AmI services cannot be provided by individual service components and, in several cases, their coordination cannot simply occur via indirect coordination mediated by the devices. Rather, they should sometimes rely on explicitly orchestrated activities of several service components, each of which in charge of controlling different portions of the environment. Simply said, AmI service components (as agents) have to live in a multiagent system. Moreover, it is expected that the interaction patterns of service components are dynamic and flexible, to tolerate dynamics of the environment and to support situation-awareness (i.e., a camera being broken or being substituted by other sensors).

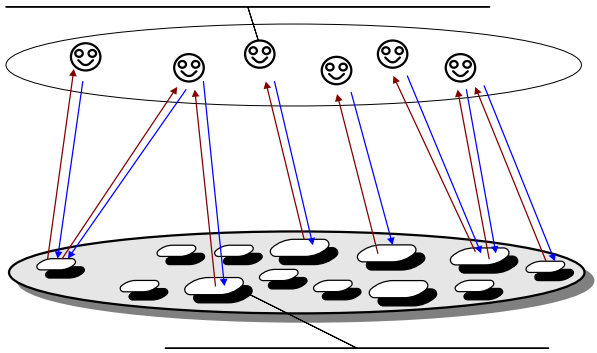

Figure 2. A general AmI infrastructure.

Besides sociality and autonomy, agents can provide another interesting characteristic, which allows them to migrate through the hosts of a network: *mobility*. In AmI, mobile agents can migrate among sensors to get local data; moreover, mobility can be exploited to manage agent updates while the system is running. An AmI system should never stops its execution, and thanks to mobile agents, it is possible to send an updated agent to the device or host it will run on, without stopping the whole system.

As an additional consideration, embedding AmI services into agents enables also a good modularity: providing new services means that new agents must be "installed" into the environment, without affecting the older ones. Please note that a service could not require only a single agent, but several agents, producing a multiagent organization.

3 Agent-Based Ambient Intelligence in LAICA

The aim of the LAICA project is to experience AmI at a urban level, enabling users to exploit digital services, with a current main focus on security and safety issues. The project is ongoing since a year and – at the time of writing – the first service demonstrators are being tested on the field.

3.1 The LAICA Infrastructure

The LAICA urban scenario involves several components, of different nature and with different computational capabilities. In particular, the project focuses on the use of traffic lights, video cameras, motion sensors and dark vision sensors (e.g., based on infrared or thermal). More than one hundred cameras and two dozen sensors some have been so far deployed on major city parks and on walking subways.

All data and streams produced by the devices automatically flow to a centralized *database*, which is in charge of logging all available information (as required by Italian laws). The above database is mainly

accessed by an *operating central*, which is in charge of managing the entire LAICA environment, and of making non-sensible information publicly available through the Web portal (for instance, to avoid violations on privacy laws, data streams coming from cameras are filtered so as to cover faces with "smiles" before forwarding them to the Web portal). Beside this, the infrastructure is mostly a distributed one.

All the distributed sensors of the LAICA infrastructure are either computer-based or coupled with an embedded PC, and thus can locally host the execution of data processing algorithms, AmI middleware components, and associated agents. In some cases, multiple devices are associated to the same PC (e.g., in the case of a set of co-located cameras pointing in different directions).

The data sensed by the devices undergoes a preliminary in-loco processing, with the aim of extracting relevant information. For example, *image-processing* techniques (as studied and implemented in a specific project workpackage [Cucchiara et al., 2005]) are exploited to detect presence of humans in parks and subways and then to track their positions, as well as to detect the specific type of a vehicle and its plate in a street. These pre-processed data, as well as raw data streams, are then made available to the middleware level.

The *middleware* is the glue component, in charge of dealing with the heterogeneity of the involved components, converting incoming and outgoing data. Moreover, the middleware includes the basic services to support agent execution, i.e., coordination logic and protocols, and discovery and event services.

AmI agents, which are scattered along the whole environment, may belong to two main classes: *sensor agents*, in charge of managing information (possibly pre-processed) acquired by devices and of controlling them; and *effector agents*, able to perform active actions on the ambient they live into. Usually, sensor agents are directly attached to an environmental device or sensor, to perform a computation locally to the device itself. This is possible because, in the LAICA environment, each device/sensor has a computational unit able to support the execution of at least one agent. The effector agents, instead, can be scattered within the environment, and can move in it in order to manage and consume the data issued by sensors and sensor agents.

Please note that, while sensor agents are tied to the sensor/device they are attached to, effector agents are independent of the AmI hardware environment, and they simply consume the data issued by the AmI infrastructure, and produce, in case, other data that can be used to instrument the environment itself.

All the agents can interact with each other, in order to coordinate within the whole environment; some data related to the environment priorities and status could also be manually set by humans.

3.2 The LAICA Agent-based Approach

All the computer-based devices of the LAICA infrastructure are characterized by hosting the LAICA middleware. The middleware is developed in Java with the J2EE technology and has a distributed nature, being associated to each computer-based device of the infrastructure. Besides dealing with data heterogeneity, the middleware is in charge of providing the basic services to support agent execution (e.g., lifecycle management, discovery, and event services) [Cabri et al., 2005].

The choice of implementing a new agent-based middleware infrastructure rather than exploiting one of the many available may appear odd. However it derives from various factors. First, we needed a very light-weight reactive agent model and from the need of enforcing a simple event-based interaction models, whereas most available middleware infrastructures (e.g., JADE [Belifemmine et al., 2001]) rely on ACL messaging and requires quite heavyweight infrastructures. Second, the actual complexity in implementing AmI agent-based services in LAICA derived mostly from the need of properly capturing and analyzing a large amount of different and possibly heterogeneous data streams, rather than on the internal complexity of agent behaviors. Therefore, during design and implementation, we felt more comfortable in defining our own kernel and middleware for agents, rather than in integrating complex data analysis algorithms into an existing system. As arguable as such as choice can be, our experience suggests that either available agent infrastructures are not flexible enough, or that not enough attention has been paid at their integration in an existing computational environment.

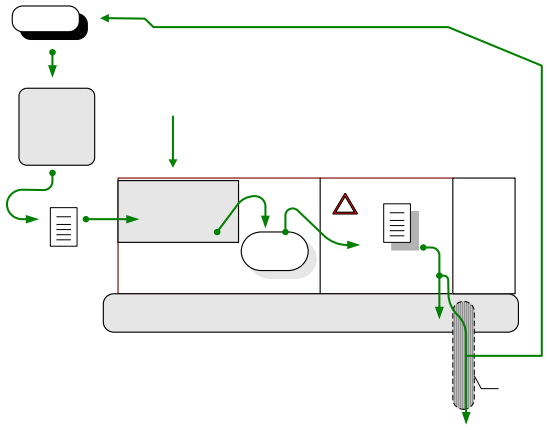

Figure 3. The general structure of a sensor agent.

From an architectural point of view, each sensor agent in LAICA has the general reactive structure shown in Figure 3. When a sensor devices issues data, that can be processed by a *device data pre-processing* module, that is in charge of applying any specific data algorithm (e.g., vision algorithms). The output, that is still raw data, is then passed to the sensor agent itself (the big rectangle in the figure). The incoming data to the sensor agent can be heterogeneous, depending from the kind of sensor and pre-processing logic. For instance, it can be an XML file, a text file or a binary stream.

The incoming raw data is initially parsed by an *input filter*, which converts it to a suitable format. The result passes through the *decision logic*, a module that, study the data with the aim of performing analysis and decision making. For instance, if the decision making parses data related to the path followed by a person and recognizes (based on actual speed) that it is a person running, it can start paying attention if data associated to another person running arrives, to identify a dangerous situation (i.e., a theft trying to catch a victim).

The results are always passed to the *event generator*, a module in charge of preparing the data for the notification to the AmI environment. LAICA recognizes two kinds of data: events and warnings. Events represent something is happening in normal conditions, while warnings represent particular (e.g., dangerous) situations. Events and warnings are then issued to the rest of the AmI environment, thus other agents (sensor and/or effector) can exploit such data to become aware of what is happening in the environment itself. A specific class of command events can be used by the sensor agent to directly control a device (e.g., change the timing of a traffic light of rotating a brandable camera). While events are issued during normal conditions, warnings require an elaboration on a set of events to detect a critical condition. Moreover, the generation of warnings depends on a few contextual information, which can be provided either by users or by other agents.

Events (and warnings) are issued by each sensor agent, depending on the decision logic, and following two parallel paths: the first is through the *storage module*; the second is through an *event channel*. The storage module is in charge of storing the event into a centralized relational database (Oracle), for further evaluation (e.g., statistics). The event channel is in charge of notifying events as soon as they happen. Please note that events can also be provided as input for a sensor agent, which can process them as commands or information about the environment status. For instance, an effector agent can issue events to activate (or change the working mode) of a sensor agent in order to acquire much data.

In the LAICA AmI environment, each agent interested in the data issued by a sensor agent must register to the corresponding event channel. When an event is issued, the registered agent is notified of such event, and thus can analyze the information. Agents are free to register and remove from each event channel, thus the environment can reconfigure dynamically. Sensor agents can show themselves to the environment through the *discovery and sync.* module, that is in charge of notifying the remaining part of the middleware of the presence of the sensor agent. Besides this, the discovery module can be exploited to find out other agents, that can help the sensor agent performing its computations, or simply to discover other similar sensor agents which this one should collaborate with.

Please note that the sensor agents, even if reconfigurable and changeable at run-time (in terms of data filtering and decision logic), are always fixed in the LAICA AmI infrastructure. This means that they are attached to a sensor, and their work will always depend on the collaboration with such sensor. On the contrary, effector agents can join and leave the AmI environment dynamically, can exploit services provided by the middleware and the sensor agents, and can rely on even more complex internal architectures (e.g. BDI) to implement more intelligent part of the environment. Also, effector agents can be associated to users and can provide to adapt service functionalities to users' wills. In fact, each user can join the environment by means of a device, for instance a PDA, endowed with its own effector agents that, when joining the environment, will be able to interact with already existing agents. For instance, if the user comes back from office, the effector agents of her PDA could join the environment and, exploiting the data provided by the sensor agents, find out the faster way to home with respect to the traffic jam, the weather, etc. Another user, on the contrary, could be waling in the park and, through a kiosk, search for a specific shop. The effector agents could suggest to the user a path to the shop and, at the same time, inform the shop that a customer is coming to buy a specific good. In this case, the effector agents initially belongs to the kiosk, but are then bound to the user that is requesting the service, and thus will behave in a way to make the environment to be perceived as intelligent.

Starting from the above considerations, it should be clear how the middleware and the sensor agents do not represent all the intelligence behind the LAICA environment, since it is mainly embedded in the effector agents, which are tied to each user and to her wills.

4 LAICA at Work

Services and applications within LAICA can be developed as a set of effector agents that exploit the services provided by the sensor agents, or can be directly embedded into the decision logics of sensor agents. The following subsections show the above two approaches by means of two service examples that have been – at the time of writing – already tested on the field with success.

4.1 Tracking a Stolen Car by Coordinating Sensor and Effector Agents

An example of AmI application that exploits the services provided by the sensor agents and requires coordination with the effector agents is the one for the track of a stolen car. Once the operating central emits the number plate of a stolen car, the middleware informs each distributed part that can "see" it (i.e., the cameras distributed in various streets of Reggio Emilia).

The sensor agents associated to those cameras are in charge of analyzing the pre-processed data of the camera (which, via image processing techniques, can recognize the plate number from the image) and, upon recognize of the stolen number plate, issue an event (through the event generator) to notify which car is currently in its sight. Effector agents catch these kinds

of events and provide both to properly alter urban police and to activate then other sensor agents.

As shown in Figure 4, there is an effector agent that works as a listener for events issued by one (or more) sensor agent. Such events include the number plate of the car that is currently in the sight of the camera, so that the effector agent can query a database to compare such number with those identified as stolen cars. Depending on the match of this last operation, the effector agent issues an alarm for the police (i.e., send the alarm both the operating central and – via SMS – to in-service policemen), and can instrument sensor agents of other cameras to keep the car in their sight.

In this kind of application, the logic is embedded in the effector agent, that starting from the event issued by a sensor agent can coordinate and instrument the whole environment in order to track the stolen car.

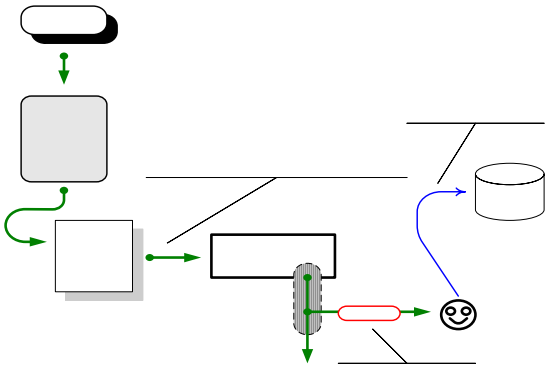

Figure 4. Data flow for recognizing a stolen car.

4.2 Identifying Odd Human Behaviors: Decision Logic in Sensor Agents

In some cases it is more efficient to embed all the needed computational logic directly in the sensor agents. These are the cases of computation that are routinely done and that affect only one (or a few) localized sensors, without requiring information from the whole environment.

An example of such application is that for the discovery of strange human behaviors. For example, it is possible to notice and track a person routinely moving around a monument, or near the children sandbox.

The data-preprocessing of cameras and sensors issues an XML file (see Figure 5 for an excerpt) that contains the information related to the person they are observing. In particular, the list of `path` tags provides the coordinates of the person in the scene.

The decision logic of the related sensor agent receives the log of the related camera, appropriately converted, and starts analyzing it for each person. Each time the sensor provides updated information for the same person, the decision logic check such data against the one already in, in order to see if the coordinates of the person are still in the same limited interval. If the list of coordinates remains in the same interval for a while, depending on a user-defined threshold, the sensor agent can issue a specific event that informs the environment of the situation (i.e., there is a persons standing nearly still an important monument).

```
<?xml version="1.0" ?>
  <tOut>10:12:00</tOut>
  <d>60</d>
  <pId>1</pId>
  <hm>0.060776</hm>
  <hsd>0.000261</hsd>
   ...
  <path>155 93</path>
  <path>155 93</path>
  <path>155 94</path>
  <path>156 94</path>
  <path>155 93</path>
  <path>154 93</path>
  <path>154 94</path>
  <path>155 93</path>
  </pers>
```

Figure 5. An excerpt of an XML document for a person's track.

5 Related Work

The agent oriented paradigm has been already exploited in the AmI scenario; nevertheless, as far as we know, no experiences are reported on using agents for urban-scale AmI services.

In [Sashima et al., 2005], each human being is represented by an agent; this approach exploits also the concept of role within a human society, reducing so the unpredictability of the AmI scenario. Even if interesting, this approach has a lack in the mapping of human roles and agent ones, that could not be enough accurate or could even be impossible.

Satoh's approach [Satoh, 2005] exploits mobile agents capable of interacting with an environment enriched with RFID tags and embedded devices. RFID tags can be used to localize users and devices, and enforce location-aware computing. User services are realized by mobile agents (provided by the MobileSpaces infrastructure [Satoh, 2000]) that can spread across the network in a location-aware fashion in order to retrieve available environmental data and to provide users with personalized context-aware services. Although this approach is more oriented to personal user services rather than (as our approach) to general users services, the enforcement of location-aware models would be worth exploring in LAICA.

KIMURA [Christensen et al., 2004] is an improved office environment that aims at monitoring and supporting human activities and collaborations through a set of agents. Agents, via sensors and monitoring of computer actions, can recognize user activities, which are then represented in a digital form as *working contexts*. These contexts, other than being possibly graphically rendered on users' displays, can act as the basis of inter-agent interactions. Via a blackboard schema, agents can put and pop messages and information, communicating to each other. Although serving a totally different (and to some extent less general) goal, the possibility of recognizing and coordinating user activities may have some social relevance in an urban scenario such as LAICA.

The SALSA middleware has been proposed in a

Device

Device data pre-

healthcare scenario for supporting AmI through autonomous mobile agents [Rodriguez et al., 2004]. These middleware shares some key ideas with our approach, by exploiting agents as entities capable of interacting with the devices of the AmI infrastructure and with each other, but the focus is mostly in providing context-aware information to visitors and alerts to doctors, rather than in providing coordinated services.

The GAIA [Roman et al., 2002] middleware is a complete framework for ubiquitous computing applications. Relying on active blackboards associated to specific locations of an environment, it can be exploited in AmI applications by acting as a media to coordinate the activities of various applications agents and devices. Nevertheless, the system takes into account only the possibility for localized interactions, without considering distributed application agents on the large scale. Thus, although the concept of active spaces may be of some interest to locally coordinate activities even in LAICA, it cannot represent a general solution for AmI services in urban scenarios.

6 Conclusions

This paper has focused on the use of agents in ambient intelligence scenarios, reporting the experience of the authors in the design and development of agent-based ambient services in the city of Reggio Emilia, an activity performed in the context of the LAICA project.

Starting from a general model, where agents are exploited to deal with data from the ambient infrastructure, a concrete implementation has been described that exploits agents at two levels. The whole environment is instrumented by a middleware that exploits the distributed sensor agents to be aware of everything is happening in the environment.

We are aware that the current ambient infrastructure deployed within LAICA is not yet very rich, which also limits the classes of agent services that can currently provided within it. Still, we believe that our solution, by building on the very foundations of the agent paradigm and directly mapping its abstractions into a concrete software infrastructure, will prove effective even when the city of Reggio Emilia will be further enriched with sensor networks, computer-actuated traffic lights, Wi-Fi meshes, and RFIDs.

References

[Belifemmine et al., 2001] F. Belifemine, A. Poggi, G. Rimassa, "JADE - A FIPA2000 Compliant Agent Development Environment", *Proc. of the 4th ACM Conference on Autonomous Agents*, ACM Press, June 2001.

[Cabri et al., 2005] G. Cabri, L. Ferrari, L. Leonardi, F. Zambonelli, "Connecting Ambient Intelligent Components via Dedicated Middleware: an Agent Approach", *Workshop on Ambient Intelligence - Agents for Ubiquitous Environments*, Utrecht (NL), July 2005 (informal proceedings).

[Christensen et al., 2004] H. B. Christensen et al., "Architecture Presentation: Experiences from Pervasive Computer Projects at Computer Science Department – University of Aarhus", *Center for Pervasive Computing, Report Series*, CfPC-2004-PB-1, revision 1.4, 2004.

[Cucchiara et al., 2005] R. Cucchiara, A. Prati, R. Vezzani, "Ambient Intelligence for Security in Public Parks: the LAICA Project", *Proceedings of IEE International Symposium on Imaging for Crime Detection and Prevention*, London, UK, pp. 139-144, June 2005.

[Estrin et al., 2002] D. Estrin, D. Culler, K. Pister, G. Sukjatme, "Connecting the Physical World with Pervasive Networks", *IEEE Pervasive Computing*, 1(1):59-69, Jan. 2002.

[Hagras et al., 2004] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, H. Duman, "Creating an Ambient-Intelligence Environment Using Embedded Agents", *IEEE Intelligent Systems*, 19(6):12-20, Nov.-Dec. 2004.

[Jennings, 2001] N. R. Jennings, "An agent-based approach for building complex software systems", *Comm. of the ACM*, 44(4):35-41, April 2001.

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

[Riva et al.] 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

[Rodriguez et al., 2004] M. Rodriguez, J. Favela, A. Preciado, A. Vizcaino, "An Agent Middleware for Supporting Ambient Intelligence in Healthcare", *Workshop on Agents Applied in Health Care*, Valencia (E), August 2004.

[Roman et al., 2002] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, K. Nahrstedt, "A Middleware Infrastructure for Active Spaces", *IEEE Pervasive Computing*, 1(4), Oct-Dec 2002.

[Satoh, 2000] I. Satoh, "MobileSpaces: a Framework for Building Adaptive Distributed Applications Using a Hierarchical Mobile Agent System", *Proc. of Conference on Distributed Computing Systems, ICDCS*, pp. 161-168, IEEE Computer Society, 2000

[Satoh, 2005] I. Satoh, "Mobile Agents for Ambient Intelligence", to appear in *Postproceedings of International Workshop on Massively Multi-Agent Systems*, Lecture Notes in Computer Science (LNCS), Springer, 2005

[Zambonelli et al., 2003] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing multiagent systems: the gaia methodology", *ACM Transactions on Software Engineering and Methodology*, 12(3):417-470, July 2003.

[Sashima et al., 2005] A. Sashima, N. Izumi, K. Kurumatani, "Social Role Awareness for Ambient Intelligence", *Workshop on Ambient Intelligence – Agents for Ubiquitous Environments*, Utrecht (NL), July 2005