

Implementation and Evaluation of a Location-Aware Wireless Multi-Agent System

Ignacio Nieto Carvajal, Juan A. Botía Blaya, Pedro M. Ruiz, Antonio F. Gómez Skarmeta.

University of Murcia, Spain.

Abstract. Determining the physical location of the users of a multi-agent system is essential if we want it to provide them with context-awareness services. A context-aware system can answer to the requests of its users in a more effective way, being able of consider the conditioners that affect them. Here we present SMAUG, a multiagent context-aware system aimed at managing the university tasks of tutor and pupils. We discuss several alternatives to obtain context-awareness and propose a concrete solution to this subject.

1 Introduction.

In the development of a multiagent system, several factors must be considered. One of this factors refers to the information the system will handle from the user's environment. This information will allow us a better interaction with the user. Perhaps the most important information we can obtain from a system that interacts with the people in a concrete area is the location of the user. This information allow us to select the appropriate services to offer to our users, give them location capacities or offer them specific resources from an area. Nevertheless, implementing context-awareness in a dynamic, open and platform-independent system is not a trivial thing. In this paper we analyze several methods to obtain it, and expose the concrete method we have implemented in our multiagent system: SMAUG.

SMAUG is a multiagent context-aware system that allows tutors and pupils of an university to fully manage their activities. SMAUG offers its users with context-aware information from their environment and also gives them a location service to physically locate every user of the system.

The remainder of this paper is organized as follows. Section 2 introduces the SMAUG system and gives a complete explanation of its motivation, objectives and structure. In section 3 we propose, analyze, criticize and compare several alternatives we have studied in our search for context-awareness. Section 4 exposes the concrete implementation we have finally decided to adopt. Last section 6 presents some conclusions and proposes a frame for future works.

2 The SMAUG System.

SMAUG is aimed at setting an ambient intelligent environment, concretely destined to university surroundings. SMAUG development follows the GAIA methodology[8], thus seeing the system as a set of entities we call agents. These agents, as said before, are independent, responsive and communicative, and can interact, cooperate and even compete between themselves.

2.1 Analysis and functionality.

Every agent makes a certain function in the system, fulfilling a functionality we call (following the GAIA nomenclature) a *role*. A role is composed by a set of behaviours taking a concrete function. There are two main users of the system: tutors and pupils, so the two main services provided are:

for tutors:

- consult and modify their own lectures, consult other tutor's lectures.
- physically locate other pupils and tutors in the building.
- search for tutors to talk about a scientific article.
- search for pupils interested in a final-year project.
- set a physical chat with people searched in the previous points.

for pupils:

- consult any tutor’s lectures. Obtain automatic information about a tutor’s lecture change.
- physically locate other pupils and tutors in the building.
- search for tutors to resolve a educational enquiry.
- set a physical chat with a tutor searched in the previous point.

Thus, we will have two "real" agents (those who are ruled by a human being): the tutor agent and the pupil agent, and some "virtual" agents (that exist only within the telematic system): the BBDD agent (that allow access to the databases) and the locator agent (that allows the physical location of the agents). We will distinguish between *active* roles, which are always waiting for message arrivals, and *passive* roles, which are inactive and only get up due to a user petition. The user will interact with the system by means of a graphical user interface, configured to adapt itself to the boundaries of the device that is being used. Thus, in figure 1, we can see the SMAUG system running on a Sharp Zaurus SL-5500 PDA under linux.

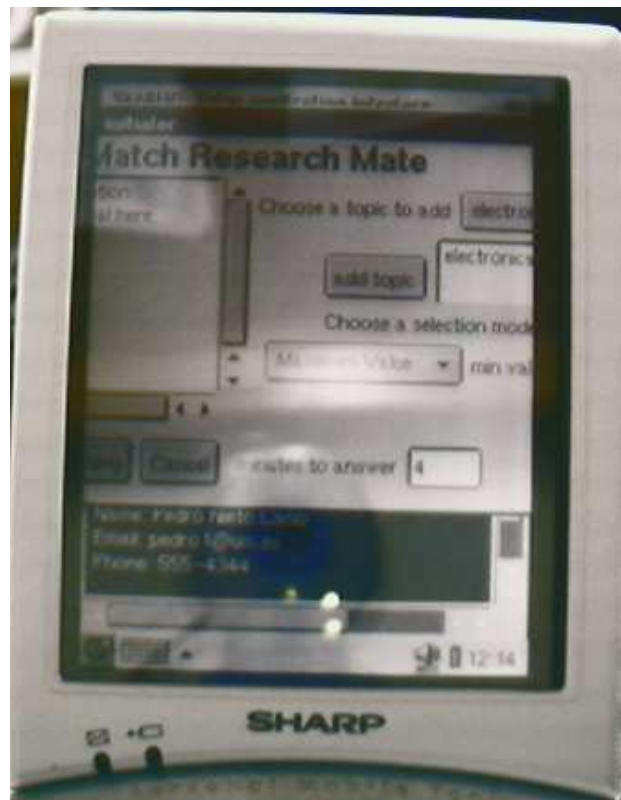


Fig. 1. User interaction with SMAUG running on a PDA.

2.2 Design and interactions

SMAUG adapts also to the FIPA specifications, to the particular platform architecture and communication structure and interaction. SMAUG interprets FIPA by means of the JADE platform, which provides an open source, FIPA-compliant, Java implemented agent system. Communication one-to-one between SMAUG agents is done by means of the FIPA-Request interaction, while one-to-many communication is done by means of the FIPA-ContractNet interaction. Figure 2 shows a typical SMAUG one-to-many interaction, precisely the interaction started by a tutor that looks for other tutors to discuss a scientific topic.

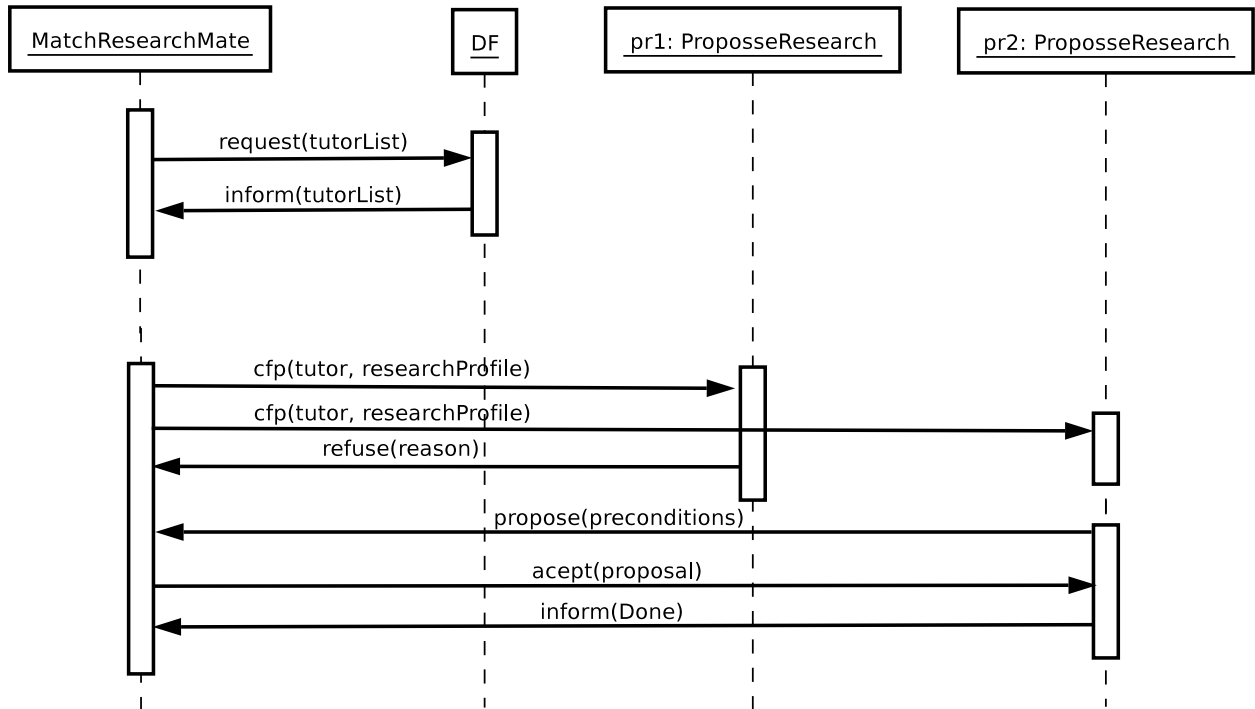


Fig. 2. SMAUG interaction example: MatchResearchMate.

2.3 System environment.

SMAUG works on certain environment. It requires a wireless infrastructure to properly offer a ubiquitous surrounding, connected to a common network containing all the servers required by the system (like the database server or the location service). SMAUG works over a huge amount of computer devices, such as workstations, personal computers, laptops, PDAs, and even less powerful devices like for example mobile phones. This requires high level of compatibility and adaptability from SMAUG.

2.4 Information manipulation: the system ontology

SMAUG works with data of the real world. In our case, SMAUG treats information such as tutors and pupils, lectures, dates, rooms, and so on. This requires the system to be able to handle, process and transmit this information. This is done by means of an ontology. In our system, the ontology is generated by means of the Protégé beangenerator utility, and translated into Java classes. The agents interpret and transmit this information cleanly, being able to manipulate real world information, thus allowing us to refer to the users context. The ontology carries the user context information. Changing the ontology will allow us to change the specific context our application is moving on. Figure 3 shows the ontology we use for the university application of SMAUG.

One of the main concepts of our ontology is the *context*. A context contains some relevant information of the world that surrounds the system. The main objective of our context is to relate entities of the ontology and integrate the skeleton for the context-awareness of the system. There is a wide tree structure of contexts, established mainly over two dimensions:

- The *persistence* of the data. In this dimension we distinguish between physical contexts (those which rarely changes, like the subjects assigned to a tutor on a year) and situational contexts (those which changes frequently, like the free seats of a classroom).
- the *topic* of the data. We have information about the location of a person, the tutoring hours assigned to a tutor, the resources of a laboratory, etc. . .

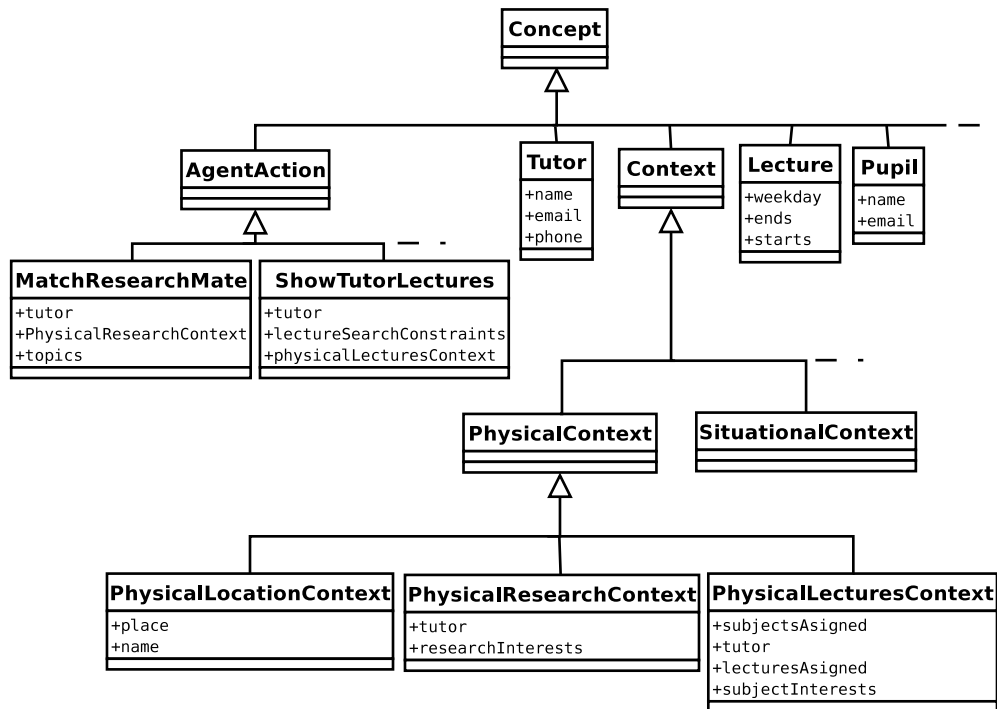


Fig. 3. Ontology of an university environment used by SMAUG.

This contextual information is stored intrinsically and extrinsically. Physical context information (such as pupils' names), mainly static and stable, is stored in a database. Situational context information is stored more dynamically in classes of the system and managed by the agents and the system itself. All contextual information is transmitted by means of FIPA interaction protocols, mainly by the FIPA-Request[3] (one-to-one interaction) and FIPA-ContractNet[4] (one-to-many interaction) protocols, being encapsulated in form of classes and objects, so agents can handle and manage it transparently.

2.5 Selection models.

One of the services provided by SMAUG is the capacity to establish meetings between its users for diverse reasons. For example, a tutor may want to look for a pupil interested in a final-year project. This interaction is done in two steps (matching the participants and selecting a meeting date). Nevertheless, in situations where there is a huge number of users (like an university), the individual selection of individuals is not an acceptable alternative. Thus we need an automated selection method (making use of FIPA-ContractNet interactions) that allows us to highly configure our decision preferences. SMAUG solves this problem by means of *Selection models*. A selection model is a tool (encapsulated within a Java class) that allows us to do a selection between a bunch of participants given some preferences, options and characteristics. The selection models works with a set of participants interested in some topics. SMAUG includes several selection models:

- *Everyone interested* model: selects every participant with a minimum interest (configurable by the user) in the topic(s). This model is of utility when we don't have big restrictions in space or time or we want the maximum possible number of people coming to the meeting.
- *Only the most interested* model: this model is of utility when we want only one participant to be selected for the meeting (like, for example, when a tutor looks for a pupil interested in a final-year project). This model only chooses the most interested participant for the meeting.
- Majority model: this model does an analysis of the medium interest of the participants and choose only those who are over this average value. This is the usual model when we want to discuss a scientific subject and we only want to meet with people with some level of interest.

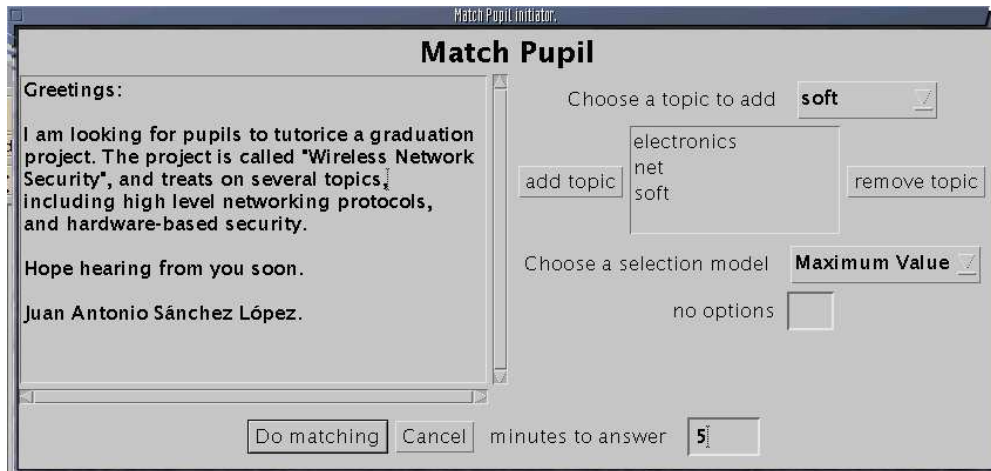


Fig. 4. Looking for a pupil interested in a graduation project with SMAUG.

- Random model: There are times when we are not really bothered about the people that comes to a meeting or we just only want certain gauging for it. In this cases we can apply the random model, thus letting the computer deciding for ourselves in a random criterion.

3 Alternatives in the search of context awareness.

In this section we review several alternatives we have studied as possible options for implementing the context-awareness of our system. Some have revealed better than others, and we include a short comparisson of their pros and cons. In every situation, we will have a mobile device (such as a PDA or laptop) equipped with a 802.11 wireless card that gets access to the network by means of one or more access points. We have several of these access points distributed in different places of the building, so we can guarantee a full area of coverage to our users.

3.1 Current Java developed libraries.

One of the first questions that arises when designing our location system is: "Is there a Java developed library that already does this?". The short answer is "No". Let's take a closer look at some of these libraries.

The first one we must check out is the Java 2 Micro Edition (J2ME). J2ME, in conjunction with The Mobile Information Device Profile (MIDP) and Connected Limited Device Configuration (CLDC) provides a solid Java platform for developing applications that run on devices with limited memory, processing power, and graphical capabilities. Unfortunately, this API works at a very high level of abstraction, and offers no way of accessing to low-level information, essential to get information about the user location.

There is a bunch of commercial solutions based on Java, such as Openwave, a SDK designed for wireless and mobile devices. This tools has serious drawbacks, such as dependency of a software supplier and costs of acquisition and maintenance. Also, this tools are not specifically thought for our system, so they contain a great level of functionality we don't need and may lack some functions we need.

There is also independent alternatives in academic circles, such as the Aura Project. Nevertheless, there seems not to be a complete academic solution out there easily and freely available.

Thus, no already designed solution seems to suit our needs. We need to implement our own solution for locating our users. This solution must integrate perfectly with the SMAUG system, being versatile enough to be reused in other applications.

3.2 Signal Strength measurement.

One of the methods we can use to obtain the location of the users is to measure the signal of the user's mobile device with respect to every access point. Each access point has an operational range in which the signal follows certain static rules. The nearer the device stays to the access point, the stronger the signal goes. Also, the structural architecture of the building affects the given signal levels. Solid objects such as walls, doors and furniture reduce the signal coverage area. Nevertheless, in a stable environment, the strength distribution of concrete zone keeps constant, and we can measure it. We can store all this information and do periodical checks to see if the situation has changed. Once we know the signal coverage of every access point, we can ask the device for the signal it receives from each.

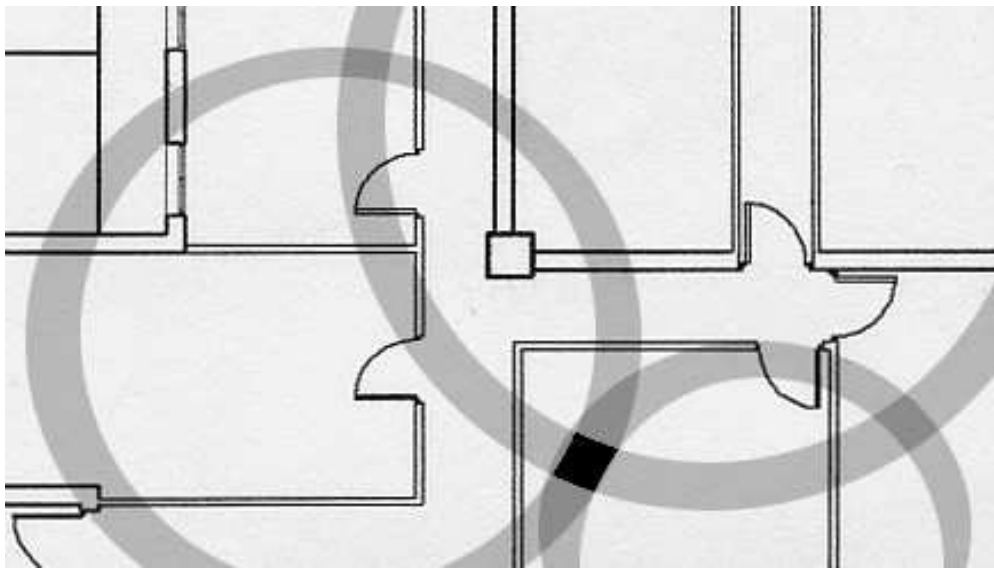


Fig. 5. Measuring the signal strength.

This approach supposes we have special hardware or software added to the devices in order to obtain this information. Usually, a concrete signal level corresponds to a circular radius centered on the access point, so we need three (in some cases two) or more access points to exactly determine the location of the user. Figure 5 shows how the signal strength of three access points can be measured to obtain the exact location of the user. Depending on the situation, two access points could give us an ambiguous location.

This method is very accurate, given enough access point and a correct initial measurement of the signal levels. Its main drawback is that it requires special hardware and software in the clients in order to obtain the signal level they receive from the access points, as not all devices and wireless cards offer this functionality. Also, changes in the signal levels of the access points (due to the inclusion of furniture or the physical change of an access point) could lead us to incorrect measurement and, therefore, to an incorrect evaluation of the user location.

3.3 Location through SNMP.

SNMP (Simple Network Management Protocol) is an application-layer protocol that facilitates the exchange of management information between network devices. It is part of the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite. SNMP enables network administrators to manage network performance, find and solve network problems, and plan for network growth. Basically, SNMP provides a set of rules that allows a computer to get statistics from another computer across a network.

A SNMP system has three essential components: managed devices, agents, and network-management systems (NMSs).

- Managed devices are network nodes that contains an SNMP agent that gets and stores information from the device, and sends or makes this information available to one or more NMSs through SNMP. In our case, the managed devices are the access point of the network. Nowadays, most access points offers some kind of SNMP support. This support is not always as extended and standardized as we would like for our purposes, as we will see later.
- The agents are usually software modules devoted to management that resides in a managed device.
- The NMSs are network nodes that collects the information from the managed devices. They process, analyze and manage this information. In this vision, our NMS will be the servers of the SMAUG location system.

We have two ways of implementing a location system via SNMP, by means of periodical queries to the managed devices or through SNMP traps.

SNMP queries.

The agent contained in a managed device collects many management information. In the case of the access points, this information can be the list of hosts associated to the access point. Thus, every time a user request the system the location of some other user, we can ask our access points for it. We can even request this information periodically and store it in a database, so we can easily recover it at will, answering more quickly to the user petition and keeping the network less congested.

The queries will be done by the servers of the system, and the information will be retrieved from the access points, so the clients will only have to request the information. This makes the operation invisible to the users, one of its main advantages. Also, this approach requires no additional software or hardware in the clients, and only SNMP software in the servers, which can be obtained freely for most platforms, as SNMP is a well known standard.

SNMP traps.

SNMP also includes a mechanism called "SNMP traps". These traps consists in messages a SNMP agent can send to the NMSs when something happens. The mechanism is similar to the event system. An event is fired in the observed device, and the agent sends a notification to the observers of this event. In our case, this events could be the attachment and leaving of the users. In such a system, when a user connects to the access point, the access point will send a trap to our server informing of the situation, so the server can store that the user is connected in an internal representation or an external database. Also, when a user gets disconnected from an access point, this will send a disconnection trap to the same server, so it can store that the user is not associated to this access point anymore.

The advantages of this method is that it don't require the periodical scanning of the network. Messages are only delivered when a user connects/disconnects. Thus, less traffic is generated. Also, we have the certainty that the user location we get is up to date.

Both approaches have, however, the same drawback. Although SNMP is an standardized protocol, not every access point supports it in the same way. The access point information is usually stored in enterprise-private variables, and, depending upon the access point, it can't be retrieved. Also, the trap system is highly vendor-specific, with specific traps that can be sent or not depending upon the concrete product. This makes very difficult the task of working with SNMP in an open system like SMAUG, where several devices must interact independently of their architecture or specific model.

3.4 Location trough authentication.

RADIUS is a widely used protocol in network environments. It is commonly used for embedded network devices such as routers, modem servers, switches, etc. It is currently the "de-facto" standard for remote authentication. It is very prevalent in both new and legacy systems. RADIUS provides a robust and centralized user administration, and its present in a way or another in most network devices, such as access points and bridges.

Concretely, most access points provides the option of authenticating the devices accessing it with the RADIUS system, by means of one or more authentication servers. If we use this authentication service, we can register in the RADIUS server the devices that access the system and from which access points. This will be very valuable not only for location purposes, but for security ones.

Unfortunately, this requires a great level of configuration at an architectural level, and a system where the users are known and rarely change. In SMAUG, we plan to have many different users that will access the system by means of an unknown number of changing devices, preventing us of using this method as a viable alternative for determining user location.

3.5 Access-point low-level strategy.

The last alternative we discuss (and the one we have chosen to implement) is obtaining the information we need from the own device. If we can access to low level information of the device, such as the MAC address of the access point we are connected to, or the name of that access point, we can use this information to access a database containing all this data and thus obtain the exact location we are in, that will lead us to the context information.

Most devices and operating systems provides a way of getting this information. For this solution to work properly, we need to access the wireless device information in a clean and transparent way. We need a common interface that doesn't bother about the way this data are obtained. In our case, this is done by supplying native libraries that are read trough the Java Native Interface (JNI). The native methods, written in low level languages (such as C) pass the low level information to the JNI, and we access this information from our Java system in a transparent way.

The main drawback of this method is the extra work required. We need to write a native implementation for every system we will work on (mainly for linux workstations, windows PCs and PDAs based on Strong-ARM processors). Once we have this, though, we can provide a high level interface in Java that access to this information.

4 Access-Point MAC Address based solution.

We have decided to implement a solution based on consulting low level information from the device in order to get the location information. In this section we explain the structure of this alternative.

4.1 Retrieving the low level information.

In the SMAUG system, we must distinguish between static and mobile devices. Static devices know their own location, as this is configured at the beginning and never changes. Mobile devices don't know this information, but can know some location metadata, such as the access point MAC address. Our Location System possess a special group of agents called "LocatorAgents". The agents from each device send this information to a LocatorAgent by means of a FIPA interaction (see 2.2). The LocatorAgent is responsible for maintaining this information and the agents are responsible for keeping their locations up to date by means of communication interactions with the LocatorAgent. If a user's agent want to know the location of another user, it queries the LocatorAgent directly.

This interaction requires a number of steps, as shown in figure 6. First of all, the agent determines if it's contained in a mobile or static device. If is contained on a static device, the information is a concrete place, configured previously. In other case, the device must call a native method written in a native language and stored in the *IwSmaug* library. All this information is sent to the LocatorAgent. The LocatorAgent then contacts with the database where this information is translated into a concrete location. The LocatorAgent then inserts this location into his internal structure, which contains an association of every agent with a concrete location, where locations are complex objects correlated with each other. The agent that sends the location executes the pseudo-code algorithm shown below:

```
InsertLocation:
    deviceType = consultDeviceType();
    if (deviceType is a mobile device)
        data = metadata (call native method
```

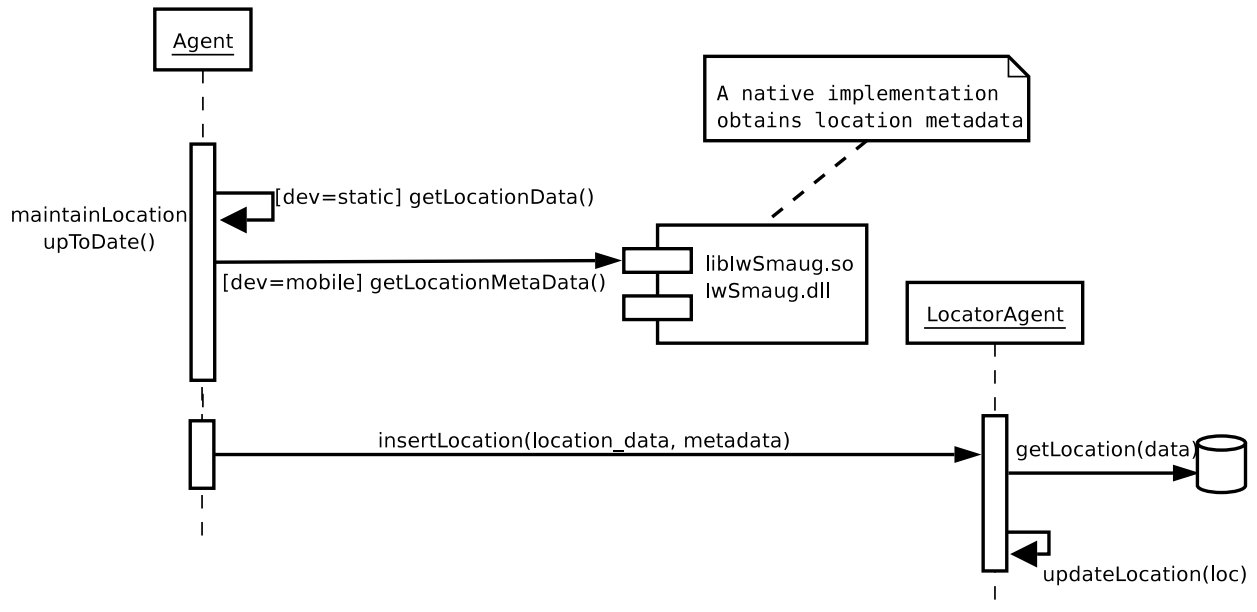


Fig. 6. Obtaining/storing the location of an agent.

```

    from libIwSmaug);
else if (deviceType is a static device)
    data = direct_location (get
        static location);
end if

send data to the LocatorAgent;

```

The Locator agent receives the petition and executes the following pseudo-code:

```

InsertNewAgent:
    data = receiveData();
    if (data is direct data location)
        location = obtainDirectLocation(
            database, data);
    else if (data is metadata about location)
        location = obtainMobileLocation(
            database, data);
    locationList.insertLocation(location);

```

4.2 The location process

Once we can access, manage and communicate the location information, we can plan a strategy for handling all this information. Our Location System consists of the stakeholders (the users' agents) and the LocatorAgents. We want to save as much bandwidth as possible, so a *polling* method is not a factible solution. Instead, our system uses a *Last known position* policy, exposed in the figure 7.

In this approach, every agent send on starting its actual position data to the LocatorAgent. The LocatorAgent calculates and stores this information along with a timestamp. If the agent is going to shut down or change its position, it informs the LocatorAgent to delete its entry or update it, respectively. When an agent sends a location query to the LocatorAgent, this searches its location

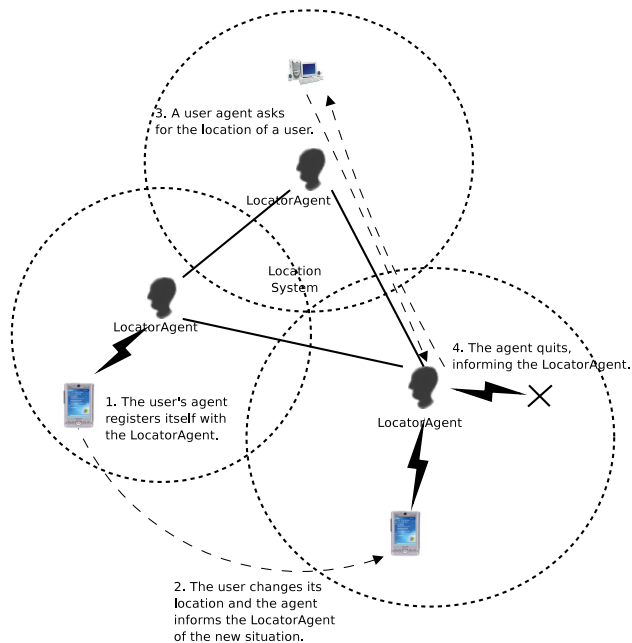


Fig. 7. Structure of the location process.

table to determine if the agent is located. If the agent doesn't belong to the table, the LocatorAgent responds that the user agent is not localizable. If there is an entry for the agent, the timestamp is examined. If the timestamp is above a certain *doubt threshold*, we are not sure if the user location is exact (it could have had problems or its machine could have broken down), so we query the agent for its location. In any other case, we are reasonably convinced that the user is on this location. This location information is of course sent back to the requester.

4.3 Sharpening the location system

Once we have the basic capacity of location through a single access point, we can go deeper and take further the accuracy of the SMAUG location system. Developing the low level information retrieving library SMAUG provides, we can establish methods for retrieving information from random access points, such as the signal-to-noise ratio. If the wireless device we are working on and the access points we possess support it, we can query the nearer access points of a device to obtain this information. With this information, we can apply a triangularisation like the one exposed in section 3.2, but without the need for installing additional hardware to the devices or using external measurement applications. Unfortunately, this information could possibly not work on every single device, but we can use it as an auxiliary tool that allows us to sharpen the location measure in the devices that support it.

4.4 Towards a middleware with location services in heterogeneous networks

Currently, SMAUG system is aimed at working on a wireless network environment, like the one we possess in our facilities. Nevertheless, nowadays there is a wide variety of network technologies available to users, and the actual tendency is to develop a roaming capacity between these networks (this is contemplated in the included in what has been known as "Fourth generation mobile networks". All of this networks (such as the Wi-Fi technology we are using, and also others like UMTS/CDMA2000) possess its own way to locate their users. Also, many other generation technologies can be used along with these.

One of the aims of the SMAUG project is the establishment of a heterogeneous system that doesn't have to rely on any particular architecture or physical device. Thus, the proposed solution needs to be shielded from the peculiarities networking equipment underneath. This means that we need to include

a global solution for all these networks into our location-awareness service. Thus, we need to build a concrete API that will serve as a high level interface for all the network location service. We will build a concrete adapter to this API for each network technology used, so we'll end up with a general interface that can be used to access the location service regardless of the particular technology in use.

Several networking technologies can be integrated into the location service:

- *Universal Mobile Telecommunications System (UMTS)*: UMTS is one of the Third Generation (3G) mobile systems being developed within the ITU's IMT-2000 framework. It is capable of supporting high-speed mobile data services as well as voice. The basic architecture of a UMTS network is similar to that of a GSM network, consisting of mobile handsets, a cellular radio network and a backbone with switches.
- *Global Positioning System (GPS)*: GPS is a worldwide radio-navigation system formed from a set of 24 satellites and their ground stations. GPS uses these set of satellites as reference points to calculate positions accurate to a matter of meters. GPS possess some advantages, like being available in the entire globe and its accuracy. Nevertheless, it has its drawbacks. GPS doesn't work inside buildings or on very populated cities, and takes a little time to get the location, due to the necessity of aligning the sufficient number of stations.
- *Wireless Fidelity (Wi-Fi)*: Wi-Fi is the short for wireless fidelity and is meant to be used generically when referring of any type of 802.11 network, whether 802.11b, 802.11a, dual-band, etc. The term is promulgated by the Wi-Fi Alliance. Our current work bases around this technology, and we are exploring in expanding it to a global network positioning and location service.

When all this network become integrated in a location service solution, we will reach a total location system where context awareness will be an integral part of the user interaction with computer devices.

5 Analytical Evaluation of the Proposed Approach

In this section, we use analytical modeling to assess how scalable is the proposed solution over a variety of operating conditions. We will use as the metric to assess the effectiveness the number of messages which are sent over the network during a certain period of time. The messages which we have considered are those used by querying agents which ask for the location of other agents, the answers given by the LocatorAgent, as well as all the location queries sent by the LocatorAgent to confirm the information in its local table and the location update messages sent by the user agents upon attaching to a new access point (AP).

5.1 Modeling the proposed solution

As we described in the previous sections, the process involved in processing a query and answering with the position of a node, requires several messages to be interchanged. First of all, one user agent sends out a query to the Locator Agent. If the information there was recently refreshed, the Locator Agent can directly answer to the querier. Otherwise, it sends out a message to contact the agent whose position is being queried, which will in turn reply with his current position. So in this case, we need two additional messages.

We will assume a typical scenario in which we have one LocatorAgent and a number of access points, each of them giving coverage to an area which we will denote as *cell* from now on. Of course, in general the number of user agents (N) will be higher than the number of access points. Otherwise, the network deployment would not be cost-effective.

We will not consider in our model the cost of queries asking for user agents which are not connected to the network. This case will just require a couple of messages (query and direct answer from the LocatorAgent) and it is not representative when considering the scalability of the proposed approach.

We use throughout this section, the notation presented in table 5.1

Let C_i be a random variable representing the time between a change of cell for the user agent number i . Each of these random variables C_i follow an exponential distribution with parameter λ_c^i . Let's assume without any loss of generality that $C_1 = C_2 = \dots = C_N$ and $\lambda_c^1 = \lambda_c^2 = \dots = \lambda_c^N = \lambda_c$.

Thus, if we define $N_c(t)$ to be a random variable representing the number of cell changes during an interval of t units of time, this random variable will obey a Poisson distribution with arrival rate equal to λ_c , whose probability distribution function (PDF) is presented in equation 1

$$P[N_c(t) = x] = \frac{(\lambda_c \cdot t)^x}{x!} e^{-\lambda_c \cdot t} \quad (1)$$

Symbol	Meaning
N	Number of active nodes
t	Duration of the time interval under consideration
λ_c	Cell changing rate (changes/second)
λ_l	Arrival rate of location requests (requests/second)

Table 1. Notation used in the model

So, the probability of the event of not having updated its position to the location agent (i.e. not having changed to a new cell) within a t times of unit interval can be easily computed as shown in equation 2, by a simple substitution in equation 1.

$$P[N_c(t) = 0] = e^{-\lambda_c \cdot t} \quad (2)$$

Accordingly, the probability of a user agent having updated its position can be easily computed as $1 - e^{-\lambda_c \cdot t}$. These equations identify the probability of the LocationAgent being able to answer directly to the queries.

Let λ_l be the arrival rate of location requests at the LocationAgent. We define again a random variable N_l as the number of queries received in a period of t units of time. Again, this random variable follows a Poisson distribution with parameter λ_l .

As we explained before, the number of required messages to answer a query will be two (if the LocationAgent has an updated position) or four (if the LocationAgent has to contact the queried agent to confirm its position). Thus, if we define $N_m(t)$ as a random variable representing the total number of messages required to answer a query in a t units of time interval, its PDF will be as shown in equation 3

$$\begin{aligned} f_{N_m}(t) &= (1 - e^{-\lambda_c \cdot t}) \cdot 2E[N_l] + e^{-\lambda_c \cdot t} \cdot 4E[N_l] = \\ &= (1 - e^{-\lambda_c \cdot t}) \cdot 2\lambda_l \cdot t + e^{-\lambda_c \cdot t} \cdot 4\lambda_l \cdot t = \\ &= 2\lambda_l \cdot t(1 + e^{-\lambda_c \cdot t}) \end{aligned} \quad (3)$$

In addition to the messages required to answer queries, we also need to take into account that each user agent will send a message to the LocationAgent whenever it detects it has changed from one cell to another. The process of detecting the change can be done locally, so that no additional message overhead is required. The expected number of messages being sent to update the location of the user agents within an interval of t units of time is presented in equation 4.

$$E[N_c(t)] = \lambda_c \cdot t \cdot N \quad (4)$$

Thus, if we define M as a random variable representing the overall number of messages during a t units of time interval, it can be defined as the sum of N_m and N_c . Thus the joint PDF is the one which is shown in equation 5

$$f_M(t) = 2\lambda_l \cdot t(1 + e^{-\lambda_c \cdot t}) + \lambda_c \cdot t \cdot N \quad (5)$$

It is important to stress that, in general, λ_l is expected to be much bigger than λ_c . This is because the former relates to the number of queries which may be high if there is a high workload, whereas the latter is directly related to the speed of the terminals. Just as a reference, a value of $\lambda_c = 0.5$ (i.e. changing from one access point to another every 2 seconds) can be considered as a very high mobility scenario, in which the nodes could be moving at around 80 m/s (assuming a radio range of 80 meters per AP). Thus, typical values for λ_c in indoor scenarios would be between 0.0125 (1 m/s) and 0.0625 (5 m/s). We offer below a more detailed analysis of the proposed approach.

In the case of assuming a periodic update of the user agent to the Locator Agent, we consider λ_u to be the rate at which location update messages are to be sent out by user agents in messages per second. In that case, the number of messages to answer a query is two all the times. One is the query message itself, and the other is the LocationAgent answer. In that case, the total number of messages can be calculated with the expression showed in equation 6. Please note, that in order to be precise, λ_c will usually take values around 1 or 0.5. Thus, $\lambda_c \ll \lambda_u$.

$$f_{M2}(t) = 2\lambda_l \cdot t + \lambda_u \cdot t \cdot N \quad (6)$$

5.2 Numerical Analysis

In this subsection, we evaluate the performance of the proposed solution based on the analysis performed in the previous subsection. We will also compare the proposed approach with the periodic update one. Unless otherwise stated, throughout this analysis we will use the values assigned to each of the different parameters as shown in table 5.2. In addition, some sensitivity analysis for the variation of the different values will be performed within this subsection.

Symbol	Meaning
N	1000 nodes
t	10 seconds
λ_c	0.05 changes per second (i.e. 2m/s with 40m range)
λ_l	10 location requests per second
λ_u	1 loc. updt. packet/second (only for periodic update)

Table 2. Values given to parameters when considered constants

Given the parameters shown in table 5.2, figure 8 shows the variation in the number of messages which need to be sent out as the number of nodes increase.

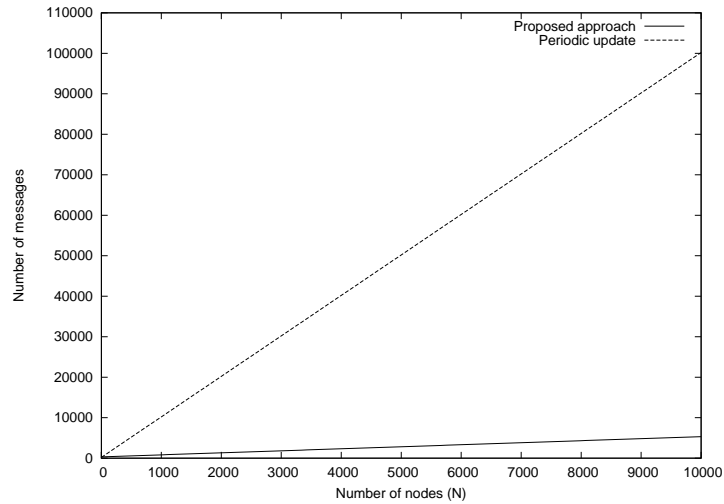


Fig. 8. Number of messages as the number of nodes increase

As figure reffig:graph1 shows, the proposed approach is much more scalable than the periodic update one. This is due because in the latter approach, each node has to update its location periodically regardless of the number of location requests. In our approach, the scalability is better because a node will only notify its location when changing from one cell to another and when someone else is asking for him.

The improvement in the scalability is also assessed in figure 9. As it is shown in the figure, the proposed approach performs much better in terms of number of messages required than the periodic update one as the number of location requests increases. The periodic update approach only is able to perform better than the proposed approach for location updates rates which are far beyond the normal use. For instance, λ_t values of 800 in figure 9, means that each of the nodes is sending almost 1 location request per second. For higher number of nodes, the advantage of the proposed approach is higher as depicted in figure 9 as well. So, we can conclude that the periodic update approach would be only suited for very small number of nodes in which there is a huge amount of location requests per node, per second. Very similar scalability gains are also obtained under the variation of the cell changing ratio, which is somehow directly related to the speed of movement. In this case, as is shown in figure ??, for the

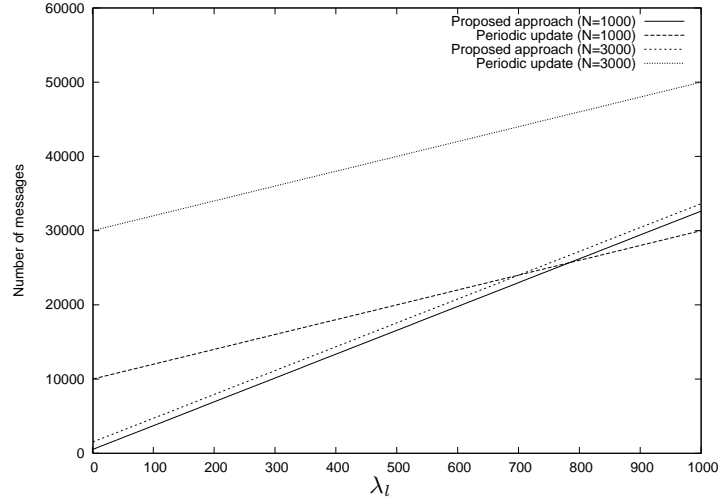


Fig. 9. Number of messages as the number of location requests (λ_l) increase

periodic update approach, the required number of messages is the same regardless of the number of cell changes per second. However, as it is depicted in the figure, our approach offers a much better performance even in high speed conditions ($\lambda_c = 0.25$ means 20m/s for a radio range of 40m for each AP).

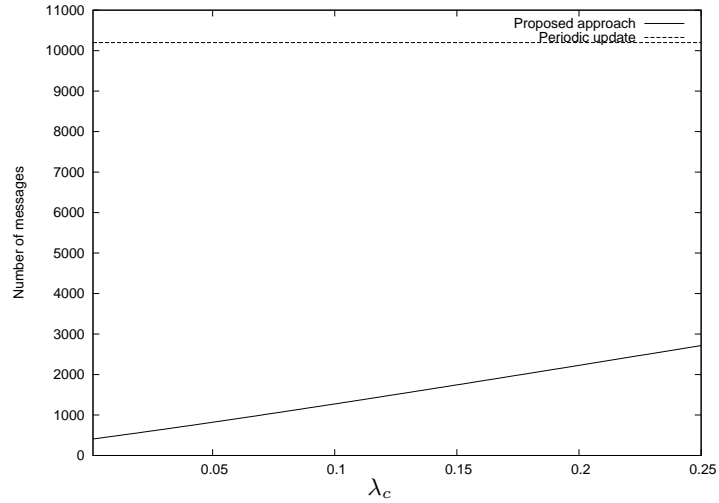


Fig. 10. Number of messages as the cell changing rate (λ_c) increase

Finally, for a better understanding of the performance of both approaches under the combination of several operation conditions, we show in figure 11 a 3D-view of the number of messages required under the variation of N and λ_c . Accordingly, figure 12 shows the same graph under the variation of λ_l and λ_c .

As figure 11 shows, the proposed approach outperforms given a fixed amount of requests per second (λ_l), the periodic update one for any N and any λ_c . This is achieved because in our approach the nodes are only requested to update their location when being located by any other agent or when changing from one cell to another.

In figure 12, we show that the proposed approach offers a better performance compared with the periodic update. The only situation in which the proposed approach does not behave better is in the unlikely event of having a limited number of extremely static devices with a very heavy load in terms

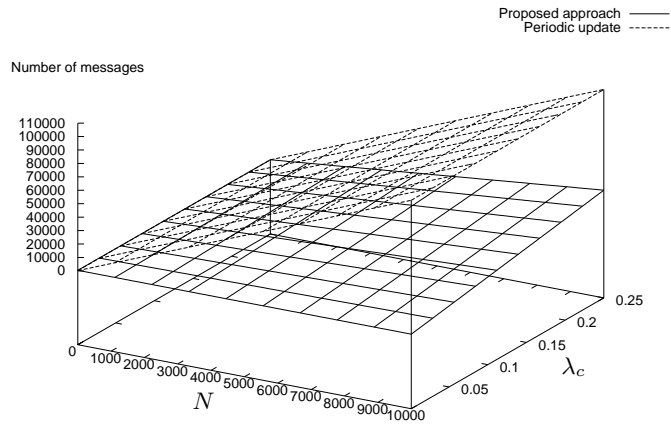


Fig. 11. Number of messages for different N and λ_c values

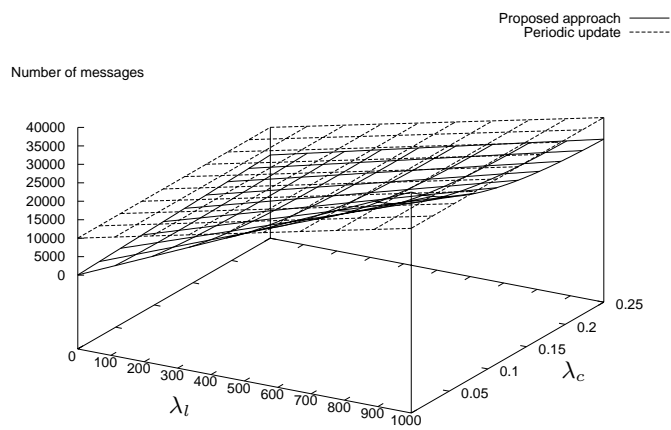


Fig. 12. Number of messages for different λ_l and λ_c values

of number of location requests per second. In addition, the higher the number of nodes, the lesser the difference between those approaches even in those unusual scenarios.

6 Conclusion and future work

In this paper we have analyzed several methods for obtaining location services for a multi-agent system. We have proposed a location management approach and we have implemented it in SMAUG, which is a multiagent system developed by ourselves. The proposed approach is based on a low level detection of the access points to which wireless devices are attached, but it does not require any changes to existing networking equipments. Although the proposed approach is integrated with SMAUG, it is modular enough to be adaptable to any other middleware framework.

We have analytically showed the scalability of the proposed approach, as well as its hability to support a high number of user-agents even when there is a high movility, and there is a heavy load in terms of location requests per second.

As future work, we aim at integrating the proposed approach with those offered by other underlying networking technologies, as well as to be able to integrate different sources of location information. As long term future work, we plan to extend this location service into a much richer context awareness service providing a much richer set of information to future applications and services.

Acknowledgements

Part of this work has been funded by the Spanish Science and Technology Ministry (MCYT) by means of the 'Ramon y Cajal' workprogrames as well as the *proyecto1* and *proyecto2* projects.

References

1. J Dale and E. Mamdani. Open standards for interoperating agent-based systems. *Software Focus*, 2(1), 2001.
2. Jacques Ferber. *Muilt-Agent Systems. An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, 1999.
3. Foundation for Intelligent Physical Agents. FIPA Communicative Act Library Specification. SC00037, 2002.
4. Foundation for Intelligent Physical Agents. FIPA Contract Net Interaction Protocol Specification. SC00029, 2002.
5. W.A IJsselsteijn G. Riva, F. Davide, editor. *Being There: Concepts, effects and measurement of user presence in synthetic environments*. IOS Press, Amsterdam, The Netherlands, 2003.
6. Michael Luck, Peter McBurney, and Chris Preist. *Agent Technology: Enabling Next Generation Computing. A Roadmap for Agent Based Computing*. Agentlink, January 2003.
7. Mark Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–100, September 1991.
8. M. Wooldridge, N. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 15, 2000.
9. M. Wooldridge and Jennings N. R. *Intelligent Agents: Theory and Practice*. Knowledge Engineering Review, 1995.