An Implemented Prototype of Bluetooth-based Multi-Agent System

Volha Bryl
Department of Information
and Communication Technology
University of Trento,
via Sommarive 14,
38050 Povo (TN), Italy
Email: volha.bryl@unitn.it

Paolo Giorgini
Department of Information
and Communication Technology
University of Trento,
via Sommarive 14,
38050 Povo (TN), Italy
Email: paolo.giorgini@dit.unitn.it

Stefano Fante
ArsLogica Lab,
IT Laboratories BIC,
Viale Trento, 117,
38017 Mezzolombardo (TN), Italy
Email: stefano.fante@arslogica.it

***Abstract*— People tend to form social networks within specific geographical areas. This is motivated by the fact that the geographical locality corresponds generally to common interests and opportunities offered by the people active in the area (e.g. students of a university could be interested to buy or sell textbooks adopted for a specific course, to share notes, or just to meet together to play basketball). Cellular phones and more in general mobile devices are currently widely used and represent a big opportunity to support social communities. We present an application of multi-agent systems accessible via mobile devices (cellular phones and PDAs), where Bluetooth technology has been adopted to reflect users locality. We illustrate an implemented prototype of the proposed architecture and we discuss the opportunities offered by the system.**

I. Introduction

Being widespread and ubiquitous, cellular phones are recently used not only as the means of traditional communication. They are also supposed to satisfy the information needs of their users, e.g. to support information search and filtering or electronic data exchange. Users equipped with mobile devices, such as cellular phones or PDAs, can form so called mobile virtual communities [1], which make possible the collaboration and the information exchange between their geographically distributed members. Such communities are inherently open, new users can join and existing ones can leave anytime. Our aim is to build a general architecture for open distributed systems that can facilitate the interaction and the collaboration among members of co-localized groups of users via their mobile devices.

We adopted Bluetooth [2] technology to connect mobile devices to servers where virtual communities based on multi-agent systems are formed and allow users to interact with one another. Bluetooth is a cheap and a widely used wireless communication technology that can connect Bluetooth-enabled devices located in a range of 100 meters.

A number of multi-agent applications to mobile devices environments have been proposed in literature. [3] presents a multi-agent system named KORE where a personal electronic museum guide provides to visitors (with Java-enabled mobile devices) information about artistic objects they are currently looking at. Information is filtered and adapted to the user profile. Bluetooth technology is used to detect the user position. In [4] MobiAgent is proposed, an agent-based framework that allows users to access various types of services (from Web search to remote applications control) directly using their cellular phones or PDAs. Once the user sends the request for a specific service an agent starts to work on her behalf on a centralized server. The user can disconnect from the network and the agent will continue to work for her. When the request has been processed, the user is informed via Short Message Service and she can decide to reconnect the network to download the results. MIA information system [5] is another example that provides personalized and localized information to users via mobile devices.

What is still missing in the above architectures is the interaction and the collaboration between the members of the virtual community. Just few proposals in the literature introduce domain-specific collaborative environments where interacting and collaborative agents act on the behalf of their users. For instance, [6] describes a context-aware multi-agent system for agenda management where scheduling agents can execute on PCs or PDAs and assist their users in building the meeting agenda by negotiating with the other agents. ADOMO [7] is an agent-based system where agents running on mobile devices sell the space on the device's screen to commercial agents for their advertisements. Agents on behalf of their users negotiate and establish contracts with neighbors via Bluetooth.

There exist a number of multi-agent platforms that can be used on mobile devices. Taking into account the limited computational and memory resources, it could be very problematic to run a multi-agent platform on such mobile devices as cellular phones. A possible solution is either to avoid running multi-agent platform on mobile devices, as for example in [7], or to use portal multi-agent platforms [8] where agents are executed not on the device itself but on the external host.

In this paper we present a general architecture based on this last option. The architecture proposes independent servers where multi-agent platforms can be installed and where agents can act on behalf of their users. Each server proposes one or more specific services related to the geographical area in which it is located (e.g. a server inside the university could

offer the service of selling and buying text books, renting an apartment, etc.) and users can contact their personal agents using their Bluetooth mobile phones. The main advantage of the proposed framework with respect to the above described architectures is that the system is domain independent (it does not depend on the specific services offered by the servers) and independent from the multi-agent technology adopted (we can use different technologies on each server).

The paper is organized as follows. Section II describes a motivating example of our system. The general architecture of the system is introduced in Section III, while Section IV provides some architectural details and describe the implemented prototype. Section V concludes the paper and provides some future work directions.

II. Motivating Example

Let's consider three places in a town: university, railway station and bar. People staying for some time in one of these places may have some common interests and needs. For instance, students at the university might want to buy or to sell secondhand textbooks, to find a roommate, or to form study groups. People at the bar could be interested in the latest sport news (especially in Italian bars), or they could just be looking for someone to chat with. Passengers waiting at the railway station may want to know some details about the trip they are going to have — what cities their train goes through, or what the weather is like at the destination point. They may want also to find someone with common interests to chat with during the trip.

Let's suppose also that people cannot or do not want to spend their time on examining announcements on the bulletin boards, or questioning people around them, or searching for the information office. They would prefer to enter the requests they have into their mobile phones and wait for the list of available proposals.

To support interests and needs of such groups of co-localized users a server is placed at each of the three meeting points. Servers can provide a certain number of services to people equipped with mobile phones or pocket computers (hereinafter referred as users). A user can have access to the services when she is close enough (depending on her Bluetooth device) to one of the three servers — at the bar, in the waiting room of the station, or at the main hall of the university.

Let's suppose that among the available services we have the following ones. University server can be used for buying and selling used books, or for looking for a roommate. At the bar sport news service is available, as well as the service which helps to find interesting people around. Railway station server gives a possibility to get information about trips (including touristic information).

Users interaction and collaboration is the base for the satisfaction of their needs. To sell a secondhand textbook, one should find a buyer and agree on the price. To find someone in the bar to chat with, one should look for the person with similar interests. Each server recreates the group of co-localized human users in a virtual community of personal agents (Figure 1) able to interact and collaborate with one another. Users formulate their requests and forward them to their personal agents.

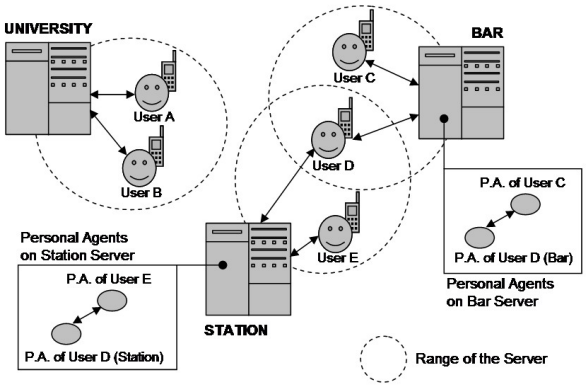

Fig. 1. Users, Servers, Virtual Communities of Personal Agents

Personal agents interacting with the other available agents (they may also negotiate, not just interact, as in the case of selling or buying books) produce results that will be sent back to the users. The main idea is to have a distributed system composed of a number of open virtual communities that evolve and act autonomously on the behalf of human communities.

III. System Architecture

In this section we describe the general architecture of the system. We start from the requirements and then we illustrate the various sub-components and their interaction.

A. System Requirements

We can summarize the requirements of the whole system in the following objectives.

- Allow the user to express her interests and choose the services she wants to access.
- Provide access to the requested services when the mobile device and the appropriate server are co-localized (i.e. the Bluetooth connection is feasible).
- Allow the user to retrieve pending results. Results should be accessible both in the case the user is still in the Bluetooth range and in the case she is out of the range.

B. System Components

The architecture of the system includes four main types of components: mobile device, PC, server and services database.

The PC component provides an interface for the user's registration to the system, for getting and choosing available services, and building requests for the chosen services. Also the pending results can be retrieved via PC. The mobile device is used to send the user's requests to the servers and to get back the results. Each server within the system provides a list of predefined services. The server runs a multi-agent platform with personal agents representing single users, a database where results are archived, and an interface responsible for establishing connections with mobile devices and PCs, and for

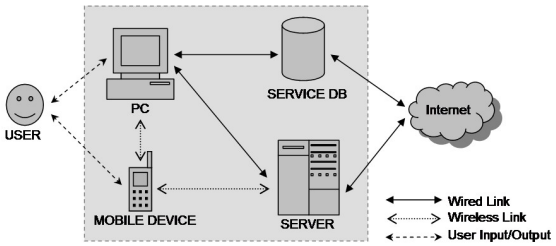

Fig. 2. Interaction of System Components

redirecting the users' requests to the corresponding personal agents. The services database, accessible via Web, contains information about all the servers and their properties, such as name, location, etc. The database provides also a description of available services on each server.

Figure 2 illustrates the general architecture of the system and the interaction among its components. Connection between the mobile device and the PC, and between the mobile device and the server is established via Bluetooth wireless communication technology.

C. Getting Access to the Services

In the following we describe how the process of getting access to the services is organized (Figure 3).

The software running on the PC allows the user to search and discover the servers and services registered to the services database. The user selects one or more services and provides information (i.e. requests) related to the use of such services. For example, using the service "Buy/sell secondhand books", the user could request to "Sell the copy of *Thinking in Java* by Bruce Eckel, printed in 1995, for the price not less than 20 euros". All the user's requests are stored in the configuration file, which is downloaded onto the mobile device via Bluetooth.

When the user with her mobile device approaches one of the servers, the software on the device establishes a connection with the server and sends the requests related to the available services. The requests are built on the base of the configuration file of the mobile device. In other words, the mobile device checks in the configuration file if the user is interested in the services provided by the server and then builds and sends the requests to the server. The mobile device stores the server's address to keep track of the contacted servers. It stores the address even if there are no relevant services on the server. This allows later the user to check the list of all visited servers and associated services, and decide to update her preferences including new servers/services in the configuration file.

D. Retrieving Pending Results

We describe now how the process of retrieving the pending results is organized (Figure 4).

The user has basically two options to get back the results of her requests. The first one is to receive them directly on her mobile device. However, this is not always possible. The

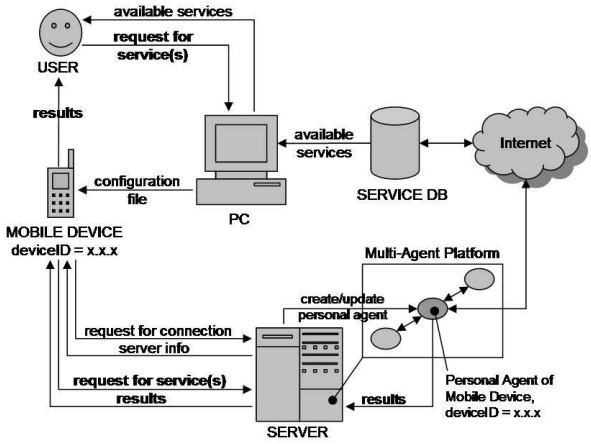

Fig. 3. Getting Access to Services

user could leave the Bluetooth area or the mobile device may not have enough memory or computational power to manage the answers (e.g. in the case the answers are a number of big files). Thus the second option is to get back the results later when the connection with the server they were requested from is finished.

Pending results can be retrieved both from the mobile device and from the PC. In the first case the mobile device has to be configured to get the pending results and has to be in the Bluetooth range of some server. For example, a student is going to spend a whole hour in the main hall of the university waiting for the next lecture, namely she will have enough time to download the results of her requests sent in the morning to the railway station server (where she bought her train ticket before going to the university). She switches on the option "get pending results" on her mobile phone, and waits for results. The mobile device sends to the university server the list of addresses of the servers the user has visited. The server establishes a connection with each server in the list, and sends the information that identifies the mobile device (e.g. its Bluetooth address) as a request for the pending results. The obtained information is sent back to the mobile device.

In the second case the user receives pending results through the PC. The student goes back home and runs the PC software that collects all the pending results obtained from the visited servers. The list of the visited servers and their addresses is transferred from the mobile device to the PC.

E. Agent Platform

Each server runs a multi-agent platform, where agents correspond to mobile devices and receive and process requests obtained from the users. We basically have a one-to-one association between agents and mobile devices (users). An agent is identified by the unique Bluetooth address of the corresponding mobile device. The same device can have many personal agents within different platforms of different servers.

When the server receives the request from the mobile device, it checks if there exists the personal agent of this

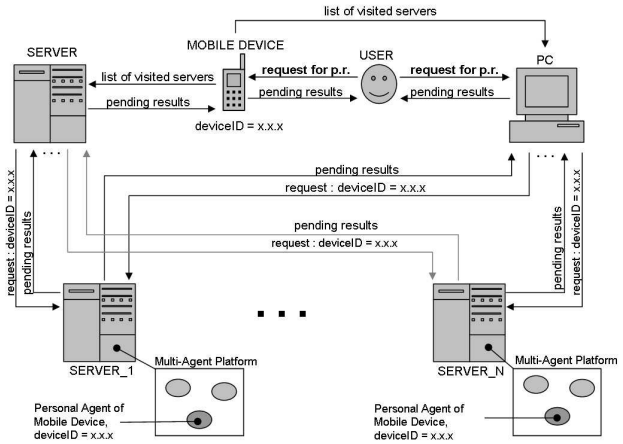

Fig. 4. Retrieving Pending Results

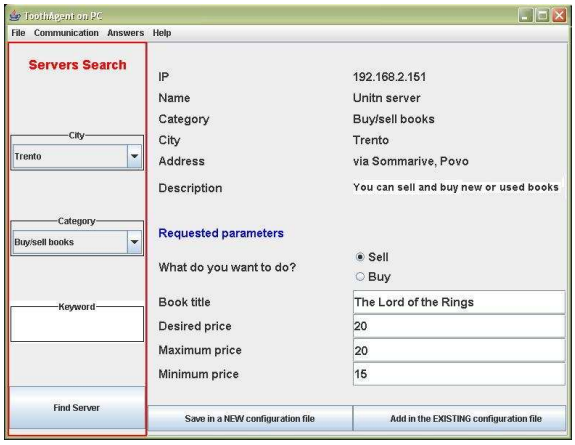

Fig. 5. Request Input Form

```
<server>
  <ip>       192.168.2.151               </ip>
  <name>     UnitnServer                 </name>
  <location> Trento, via Sommarive, Povo </location>
  <service>
    <description> Buy/sell new and used books </description>
    <parameters>
      <param>
        <paramname>  Book title         </paramname>
        <paramtype>  String             </paramtype>
        <paramvalue> Lord of the Rings  </paramvalue>
      </param>
      <param>
        <paramname>  Desired price </paramname>
        <paramtype>  int           </paramtype>
        <paramvalue> 15            </paramvalue>
      </param>
      <param>
        <paramname>  Maximum price </paramname>
        <paramtype>  int           </paramtype>
        <paramvalue> 20            </paramvalue>
      </param>
    </parameters>
  </service>
</server>
```

Fig. 6. Configuration File

device within the platform. If not, new personal agent is created. Personal agent communicates and collaborates with other agents in order to find ”a partner” which will satisfy its request. Interaction protocols and collaboration mechanisms are domain (services) dependent.

IV. Implementation Issues

In this section we present the details of the implemented prototype. Basically, the system is a first implementation of the architecture presented in Section II and focuses on a number of servers spread around the university campus (faculties, libraries, departments, etc.). Each server offers only the service for selling and buying books. We are currently working on a number of other services including services available on servers located outside of the university campus (e.g. train station, museums and places close to touristic attractions).

A. On-line registration and services selection

To start working with the system, the user has to register. She can fill the on-line registration form where she needs to put her personal info such as name, birth date, e-mail, Bluetooth address and phone number of her mobile device, and password. The registration, basically, allows the system to identify the user and the mobile device she is going to use. Password is used to access the information about servers and related services and to upload/update the user information (e.g. the user can decide to use different mobile device or just to change her data such as telephone number or e-mail address). All this information is stored in the services database. Registered users obtain the rights to download the software for the PC and the mobile device components (which are two jar files), and the XML file containing all available servers with corresponding services.

After the registration (or login), the user can start selecting services to use. Using the Java GUI interface shown in Figure 5, she can explore all the available services using filtering criteria such as server location (e.g. we can have servers located in different cities or in different places in the same city), type or category of the service (e.g. buy/sell books, exchange courses’ notes, or meet people), and keywords (e.g. books, course, etc.). The list of the selected services is managed by the PC component that allows the user to customize these services with the specific requests (e.g. title of the book to buy or to sell, the desired price, minimal or maximal price).

The list of services (with related servers’ addresses) are stored in a XML configuration file, which is uploaded via Bluetooth in the mobile device. Figure 6 shows an example for the ”sell/buy books” service.

B. Accessing the services

To access the services, the user needs to run the Bluetooth application in her mobile device. The application is written in Java and uses JSR-82 [9], which is Bluetooth API for Java. The application starts a continuous search for the Bluetooth-enabled devices in its neighborhood, and whenever it finds a server with the services specified in the configuration file, the mobile device sends the user’s requests to the server. Figure 7 shows the protocol we use for the interaction among the different components.

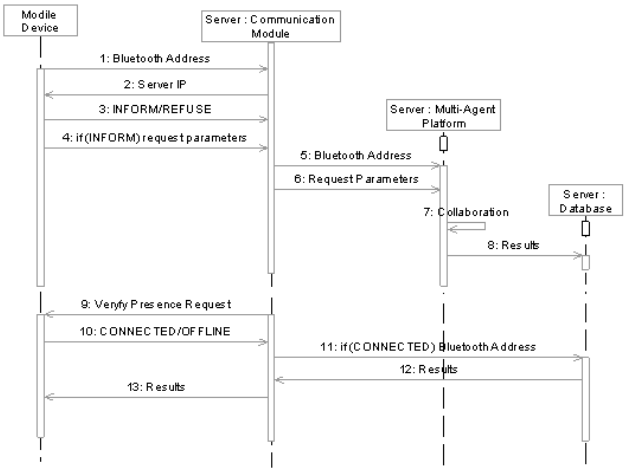

Fig. 7. Getting Access to Services

A specific communication module on the server is responsible for managing the interaction with the mobile device. It receives the list of requests from the mobile device and checks whether in the platform (running in the server) already exists a personal agent assigned to that mobile device (the Bluetooth address is used to map the mobile device with the personal agent). If there is no personal agent for the user, the communication module connects to the central services database and verify whether the user is registered to the system. Only in case of a positive answer, it creates a new agent and assigns it to the mobile device (user). Then, the communication module forwards all the user's requests to the personal agent.

Now, the personal agent starts the interaction with the other agents on the platform trying to satisfy all the user's requests. In our example the personal agent receives one or more requests for buying and/or selling books (with specified title, desired price, maximum and minimum prices, etc.). If the agent reaches an agreement with another agent about their users' requests, it can decide either to send the results back to the user or store them locally in the server database. This depends on the retrieval modality that the user has defined in the configuration file.

C. Results retrieval

Whenever a new connection between a server and a mobile device is established, the communication module sends to the mobile device the IP-address of the server. The mobile device stores the IP addresses of all the visited servers in an XML list (Figure 8-a), that is used later to retrieve all pending results. The format of the results produced by the personal agent is shown in Figure 8-b. It may contain the request identifier, the contacts (e.g. phone number) of the user interested to buy or sell the book, the actual agreed price, etc.

As discussed in Section III, the user has three different modalities to retrieve results: get the results immediately, get pending results using the mobile device, and get pending results using the PC. Each of these modalities has to be defined

```
<iplist>
  <ip>  192.168.2.148  </ip>
  <ip>  192.168.2.151  </ip>
…
</iplist>
```

(a)

```
<responses>
  <response>
    IP server: 192.168.2.151,
    Name: UnitnServer,
    Message: Buyer: Stefano Fante,
        Phone: 230-5658821,
        E-mail: stefano.fante@arslogica.it,
        Book Title : The Lord of the Rings,
        Price: 20 euros
  </response>
  ...
</responses>
```

(b)

Fig. 8. XML Formats. (a) List of IP Addresses of Visited Servers. (b) List of the Responses

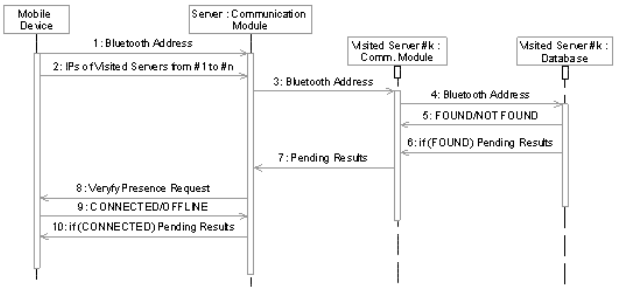

Fig. 9. Pending Results from the Mobile Device.

in advance by the user and can be changed at runtime by means of the mobile device application.

Choosing the first option, the user can receive the results immediately in her mobile device. Of course, she can receive the results if and only if she is still at a Bluetooth distance from the server. The communication module checks the availability of the mobile device and sends to it the results obtained from the corresponding personal agent.

Figure 9 shows the interaction protocol of retrieving the pending results via mobile device. Consider for example the situation in which a user is near to the server of the central library. After the connection has been established, the mobile device sends the list of IP-addresses of all previously visited servers (e.g. faculty servers, departments servers, etc.) to the library server. The communication module of the server sends then the Bluetooth address of the mobile device to all listed servers. In turn, the communication module of each server extracts from the internal database all the stored results related to the user and sends them to the requester server. All the results are collected by the communication module and finally sent to the mobile device. If the mobile device is no longer connected to the server (e.g. the user has left the library), the retrieval process will fail and the results will be cancelled (they are still available on the original servers).

Figure 10 shows the interaction protocol of retrieving the pending results via PC. The user connects her mobile device to

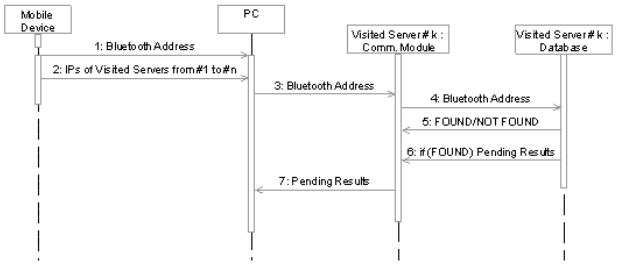

Fig. 10. Pending Results from PC

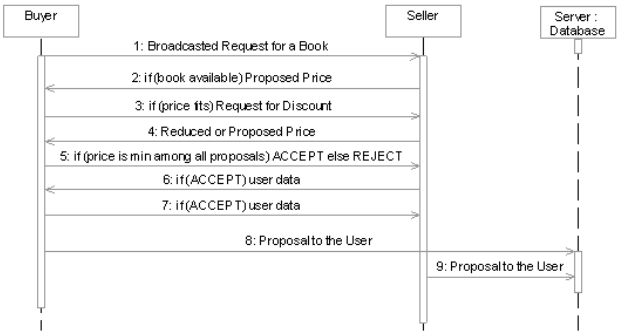

Fig. 11. Agent Interaction

the PC via Bluetooth and sends the list of all visited servers to the PC component. Now, the user can decide either to retrieve the results from all the servers or she can just select some of them. An interface on the PC allows the user to connect to the servers and then view or download the pending results.

D. Agents interaction

As we said in this first prototype we implemented just one kind of service, namely the ”buy/sell books” service. The multi-agent system has been implemented in JADE (Java Agent DEvelopment framework) [10]. The interaction mechanism is very simple. The point here is that we do not pay particular attention to the multi-agent interaction since we are mainly focused on the design and the implementation of the whole infrastructure.

Figure 11 presents the implemented interaction protocol used by the agents in the case of the ”buy/sell books” service. Buyer’s personal agent broadcasts the request of looking for a specific book (information about title, desired price, etc. are specified in the message). If in the platform there is another agent that is selling the requested book, it responds to the buyer with the price it wants for the book. If the price is greater than the maximum price specified by the buyer, the interaction continues with a discount request from the buyer agent. The seller responds either with the discounted price or with the initial proposed price (in case it does not want to give the discount). If this price is less than maximum price for the buyer, it accepts the deal. After that, the buyer and seller personal agents exchange their users’ data, form the agreed proposals and send them to the server’s database. The proposals are than forwarded either to mobile device, or to the PC as described in Section IV-C.

We tested the system using Nokia 6260 cellular phones and PC/Server equipped with Tecom Bluetooth adapter. Bluetooth communication has been implemented using Blue Cove [11] which is an open source implementation of the JSR-82 Bluetooth API for Java.

V. Conclusions

In this paper we have presented an implemented prototype where multi-agent systems and Bluetooth wireless communication technology are combined together to support co-localized communities of users. We have discussed the general architecture of the system and we have presented using the buy and sell books example some implementation issues related to the prototype we have built.

A lot of work has to be done to make the system working in a real-life environments, including the implementation of various multi-agent systems able to provide different kinds of services. We are currently working with ArsLogica s.r.l. in the development of a real scenario where to apply the system.

Acknowledgement

We thank ArsLogica s.r.l. for the collaboration and the support to this project. This research also is partially supported by COFIN Project ”Integration between learning and peer-to-peer distributed architectures for web search (2003091149 004)”.

References

[1] A. Rakotonirainy, S. W. Loke, and A. Zaslavsky, “Multi-agent support for open mobile virtual communities.” in *Proceedings of the International Conference on Artificial Intelligence (IC-AI 2000) (Vol I), Las Vegas, Nevada, USA*, 2000, pp. 127–133.
[2] The official Bluetooth website — http://www.bluetooth.com/.
[3] M. Bombara, D. Calì, and C. Santoro, “Kore: A multi-agent system to assist museum visitors.” in *Proceedings of the Workshop on Objects and Agents (WOA2003), Cagliari, Italy*, 2003, pp. 175–178.
[4] L. Vasiu and Q. H. Mahmoud, “Mobile agents in wireless devices.” *Computer*, vol. 37, no. 2, pp. 104–105, February 2004.
[5] MIA project — http://www.uni-koblenz.de/∼bthomas/MIA_HTML.
[6] O. Bucur, P. Beaune, and O. Boissier, “Representing context in an agent architecture for context-based decision making.” in *Proceedings of the Workshop on Context Representation and Reasoning (CRR’05), Paris, France*, 2005.
[7] C. Carabelea and M. Berger, “Agent negotiation in ad-hoc networks.” in *Proceedings of the Ambient Intelligence Workshop at AAMAS’05 Conference, Utrecht, The Netherlands*, 2005, pp. 5–16.
[8] C. Carabelea and O. Boissier, “Multi-agent platforms on smart devices : Dream or reality?” in *Proceedings of the Smart Objects Conference (SOC03), Grenoble, France*, 2003, pp. 126–129.
[9] JSR-82: Java APIs for Bluetooth — http://www.jcp.org/en/jsr/detail?id=82.
[10] Java Agent DEvelopment Framework website — http://jade.tilab.com/.
[11] Blue Cove project — http://sourceforge.net/projects/bluecove/.