How do I design a location-dependent application?

Filipe Meneses*, Adriano Moreira*, Rui José*
Information Systems Department, University of Minho

ABSTRACT

The generalisation of the Internet and the recent technological developments in embedded systems and wireless networks contribute to the realisation of a vision where access to information is possible at any moment and from anywhere. This is particularly attractive with information that is relevant to a specific geographic location. Applications that rely on location-based services to provide information to mobile users, or that support interaction with real devices in the user neighbourhood, are called location-dependent applications and enhance the relationship between mobile users and a specific geographic location. However, the design of such applications breaks with the existing paradigms and methodologies as the mobile devices and the wireless communication infrastructures have characteristics that are very different from those of desktop computers and wired communication systems.

The Around architecture is an open and extensible framework for location-based services that allows network services to be associated with specific geographic locations. By using this architecture applications can select services that are relevant for specific locations. Within the context of the Around project we have developed a prototype system with multiple location-based services and an application that accesses these services to provide information related to a town transportation system.

This paper reports on the design and development of this location-based application. Its design raised several new issues, going from the computational model to the interface paradigm, which are also discussed in this paper. The developed application is composed of several modules: a set of agents which are autonomous units with the knowledge necessary to select and use location-based services in a specific thematic area (e.g. transportation); an HTML output area based on a browser metaphor; and a location-context module responsible for determining the user context.

The results show that an application architecture based on a modular approach turns to be very flexible as it becomes very easy to extend its functionality by simply adding or changing the agents that deal with each location-based service.

Keywords: Location-based services; Location-based application; Mobile computing; Context-aware applications

1. INTRODUCTION

The Internet is getting bigger every day, with more web pages, more services and more information available. While the prevailing access model to the Internet is still based on a desktop PC, technological development is rapidly enabling access from everywhere, using mobile devices equipped with wireless network interfaces. The geographic location of mobile users is also becoming increasingly available through several location technologies, such as Global Positioning System (GPS) or location systems based on cellular networks. This scenario is triggering the development of a new set of services and applications known as location-dependent.

Location-based services are those services that are only relevant to a specific geographic location, i.e., that are available only to those users located inside the area of application of the services. Because the actual conceptual schema of the Internet does not support the association of services to geographic locations we have developed and used a new architecture – the Around architecture. The Around is a generic, flexible and scalable architecture for location-based services[1].

* {meneses, adriano, rui}@dsi.uminho.pt; phone: +351 253510148; fax: +351 253510250; Information Systems Department, University of Minho, Azurém, 4800-058 Guimarães, Portugal. http://www.dsi.uminho.pt/get/

Location-dependent applications are applications that provide information content or other services to mobile users that are relevant to their location. In the Around architecture, location-dependent applications rely on location-based services and on a service discovery mechanism that provides the association between the user location (context) and the services.

Several other projects associated information to physical location.

The C-MAP project[2] was an attempt to build a personal mobile system to give information to the visitants of an exposition. The system used the active badge system[3] to locate people inside buildings and some applets developed in java, running inside HotJava browser, were responsible to give the users the information for their present location.

In the Cyberguide[4] the authors built several small mobile electronic guides. Those guides were firstly thought for people visiting a building, and then extended to outdoor use. The clients had all the information locally and they showed the right information for the present location. The location of the mobile units was done using Infrared beacons. On the outside of buildings they used a GPS receiver to obtain their location. Technical limitation impaired the same system to work inside and outside the buildings.

The Guide project[5], developed at Lancaster, uses a wireless infrastructure to support the information needs of Lancaster tourists. Under this project a wireless LAN (WLAN) was installed and a specific client was developed to help tourists visiting the town, allowing them to access information according to their location. The Guide client, based on a browser metaphor, had several possibilities: get information for the present location, get weather forecast, get a map, etc. The WLAN cells strategically placed were responsible for client location: information broadcasted from the cell allows the client to know its location. A different context was associated with each WLAN cell coverage area.

The Cooltown project[6] is a platform for applications based on locations with the goal to support the correlation between the physical world entities and the web pages of those entities.

Although these projects achieved their goals, they used some specific technology, working only on a restricted scenario where those technologies were available. The Around architecture is a conceptual model to support the association between information and location, that eliminates the use of specific communications or positioning technologies.

The development of location-based applications raises many new issues, not common to applications targeted for desktop computers and that rely on wired communications infrastructures. This paper addresses these issues. In section 2 we describe the challenges that have to be faced when designing a location-based application for mobile devices. In section 3 we describe the scenario on which an application was specified for the Around project and the requirements and limitations for that application. In section 4 we report on the application development, including the choices that were made in terms of design. Some conclusions and remarks are presented in section 5.

2. THE CHALLENGES

The design of a location-based application faces several different challenges. In this section we introduce and discuss these challenges.

The computation model. One possible approach for the computation model is to develop the application as a set of servlets running on a web server and to develop the user interface as a set of web pages dynamically generated at the server. In the mobile device, only a web browser is required. With this model new functionality and new services are easily introduced since changes are only required on the server side. On the other hand, the application interface is limited by the browser capabilities and the server has to deal with the many differences between browsers used in different devices. An alternative is to divide the application between the web server and the mobile device: part of the application runs on a web server (servlets) and other parts run on the mobile device browser as applets. With this approach some extra functionality can be introduced into the application. In particular, some parts of the application dealing with the user position can run on the mobile device, possibly enhancing the user privacy. At the limit, the entire application can be downloaded to the mobile device and run in disconnected mode. However, not many browsers for Personal Digital Assistants (PDAs) support applets.

Another possible computation model relies on specific applications developed for each type of mobile device that have to be previously installed. These applications implement the user interface and have the required knowledge to deal with the network services (information services, interaction services and, eventually, positioning services). In the simplest form, the application is only able to deal with a predefined set of location-based services, and the user interface is optimised for those services. In a more advanced form, the application is able to download modules from the network and integrate them with the basic application functions: location management, user contextualisation and user interface. With this model a much more flexible user interface can be developed since it is not limited to a set of browser capabilities. In particular, this model allows, or facilitates, the support of "push mode services", that is, services that push information to the user interface. This feature is very attractive for location-based applications as it allows the application to notify the user upon changes in his/her position or to support many forms of messaging.

On-line/Off-line operation. Location-based applications do not always have to rely on a permanent connection to the network. Tourism guides, for instance, can be based on information contents stored in a local database and on a local position service such as GPS. Many currently available PDAs have enough memory to store information about an entire city. However, this approach is not appropriate for very dynamic services such as a weather forecast service or a hotel booking service.

When the application relies on data available through the network, its behaviour has to consider that a permanent network connection is not always possible. Wireless networks are less reliable than wired systems, the bit error rate is higher, the delay is usually higher and coverage is not available anywhere. Thus, the application has to deal with all these limitations and should not simple become unusable when there is no connection.

Caching of data. One possible solution to deal with the wireless network characteristics is to cache some data for future uses when there is no network connection. The cached data can include data recently requested by the user and data pre-fetched by the application (data that will probably be requested by the user)[7]. The implementation of a caching feature has to consider the amount of memory available on the mobile device, the cost of the wireless network service if a WAN is being used for pre-fetching, and the permanent user movements. For very dynamic services, caching may even be not very useful. The issue is then: does it worth to cache data? If yes, how much?

User location and contextualisation. Several positioning techniques can be used to locate a user and adapt the behaviour of a location-dependent application. In some cases the mobile device obtains the position information locally. This is the case for GPS, for some cellular based positioning techniques such as Cell Identification, and for positioning based on beacons or tags such as bar codes. In other positioning techniques the user (device) position is calculated by some external infrastructure and provided through a network service. Some positioning techniques supported by cellular networks use this approach, such as the Up-Link Time of Arrival (UL-TOA). In either case, the user position is often used as a key to access information relevant for a specific location or it is used as part of a service itself (such as a map service that displays the user position). In the Around architecture, the user position is also used to associate a location-context to the user. This contextualisation process can be implemented by the application running in the mobile device or it can be provided as a network service.

The choice of a technique for positioning and of a solution for centextualisation has a major impact on the user privacy. Local solutions preserve the user privacy while solutions that rely on network services may disclose the user location.

User interface. The interface with the user is the design of the parts of the system with which the users interacts[8]. They should be simple to use and designed in a way that is easy and efficient to work with. Design an interface is a multidisciplinary area that is not limited to computer science. Because some services demand a more complex interface (for instance, a form to retrieve users data) than others, the interface of the application should be adapted to the services used and to the device where it will run.

At present, there are a wide variety of mobile devices, each of them with particular characteristics[9]. The available memory, the processing power, the display characteristics (size and colour), the input method and the battery capacity differs from device to device. These limitations impose many difficulties in the development of a standard user interface. Mobile phones are usually very simple and present the most severe limitations. PDAs or handheld computers are more flexible, but even for these there are many differences and limitations, which include the communications capabilities and the operating system architecture. For applications that are developed using a model based on a browser, some server-side solutions are being

proposed that are able to generate the user interface "on-the-fly" depending on the device that is accessing the application. However, these translators are far from being a general solution.

Moreover, the development of an interface for location-dependent applications raises other issues. Optimised interaction with location-based services suggests the development of specific interfaces for each type of service. On the other hand, richer and friendlier interfaces can be achieved if the application is able to integrate several services into the same interface (e.g. display on a map the location of the nearest hotel that has some vacancies and that match my price range). The interface should also be able to notify the user about the availability of new data as the user moves from one location to another. And how should the application interface behave when the user is reading information that is no longer linked to he/she new location because, meanwhile, he/she moved to a new location? Should the display be updated automatically or should it signal the user that the current information is outdated? Should the same approach be used for every service?

3. DEVELOPMENT SCENARIO

The actual framework of the Internet does not have any conceptual schema to deal with location. The Around architecture is a conceptual model to support the association between the information and location, eliminating the use of one specific technology. The Around architecture intends to be a generic, flexible and scalable concept to built location-based services on the Internet.

The Around architecture[10] is based on the existence of a shared set of symbolic locations called "location contexts". There is an association between the location contexts defined and supported by the Around servers and the physical space: each location context is linked to a physical space, even if in that area there are no computer networks. Independently of what network users are connected to, they are in a certain context if they are within the physical space associated to that context. As symbolic entities, the contexts can easily represent physical spaces identifiable by humans (e.g. Library, Stadium area, Castle area, etc.). The architecture is based on Location Based Servers (LBS) and does not define any specific technique that the clients should use to obtain their present location. A contextualisation process is responsible to associate the user position, provided by a certain location system, to a location context. The LBS is responsible to keep the service registration for a set of location contexts and satisfy the services request of the clients. The clients sent their queries indicating the type of service wanted, their present location context and some optional information about searching adjacent and/or wide area contexts. As an answer to a query, the LBS return the service(s) reference(s). Another function of LBS is keeping the hierarchy of location contexts, registering information about the contexts and the relations between the contexts. The relations of "being contained in" and "being neighbour of" allow links between the contexts to be established that represent the physical reality. Relations between contexts defined in different LBS servers produce a LBS federation.

In our scenario several contexts have been defined, going from small areas inside the University Campus to wide areas in the town. Figure 1 shows the location-contexts structure defined for our demonstration.

Several location-based services were then developed and registered into these location-contexts. The *Local Description* service provides structured information about the user current location, including a textual description of the local, the postal code, useful phone numbers, addresses, etc.; with the *Weather* service a local weather forecast can be obtained; the *Maps* service provides local maps in several formats; the *BusInfo* service provides information about the bus transportation services, including information about bus stops and time tables, while the *BusStop* service was built to be used in a bus stop to give the users information on real-time about the buses arriving and bus schedules.

These services were implemented using several different technologies. The Local Description service was structured in XML files available through HyperText Transfer Protocol (HTTP). The Weather service was built through information available in different web pages. The Maps service was built using several files of different kinds made available in a HTTP server. The BusStop and BusInfo services are built with a relational database and real time information received from the buses (bus positioning). The use of different technologies showed that the system works equally well with different approaches.

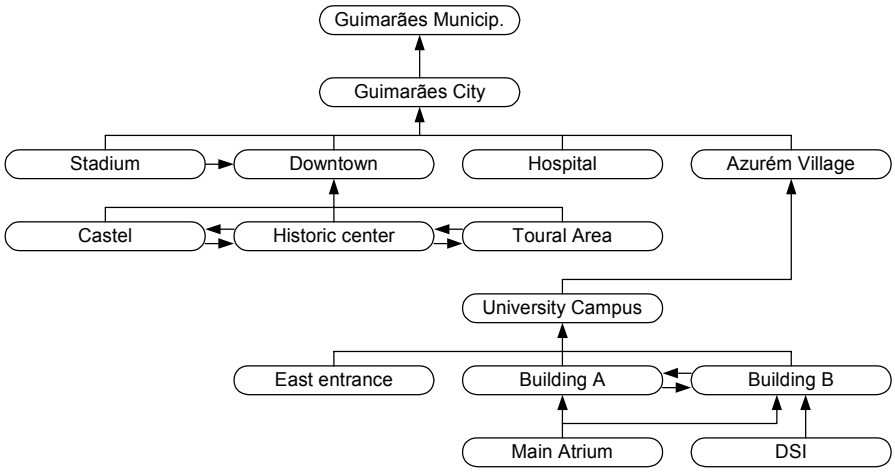

Figure 1: Location contexts for Guimarães municipality area.

All theses services were made available at various levels of the hierarchy of location contexts. Some of the services were registered in location contexts representing small areas, others were registered in location contexts representing wide areas (e.g. the weather forecast is the same to the entire town) and others were registered in small and wider areas simultaneously.

Accessing the services is simple. When the client sends a query to the LBS, it returns the service reference. With this reference the client can easily access the data, using the well known methods of that object.

A person travelling in an unknown city is perfect example of someone who needs to use location-based services to achieve information for his/her present location. With the tourist in mind, having the Around architecture and all this location-based services we have built an application to study and try to solve the challenges previously discussed.

4. APPLICATION

As discussed in section 2, developing an application to access location-based services can be a challenging task. Several decisions have to be taken before starting the development. First, we decided to produce an application that will run on the user device and it was designed for a laptop computer. A laptop computer is not the most appropriated device to use while in motion. However its capabilities are the most suited to test several possible solutions: the processing power, display capabilities and connection options makes it a very powerful solution. On the other hand, the application developed for a laptop computer can easily be migrated to a computer installed, for instance, inside a bus.

The Around architecture does not define any specific mechanism for the user device to achieve its position. For this prototype, a solution for locating the mobile device was built based on beacons. Beacons are small packets sent through the network with a string indicating the context they represent. These beacons propagate only in the network physically available in the context defined by the string inside the packet. So, different beacons, sent in separated networks, allow the use of the local area network to find out the context.

We have decided to produce an application for a well know set of services, that runs on the users device and that gets its current location by receiving beacons broadcasted by the network. Due to the lack of connectivity that sometimes happens in wireless networks, working without connectivity was also predicted.

Our application has a simple interface based on several buttons placed on to the top, a navigation area used to display some information and a major area that presents the information retrieved form the several services used. A screen shot of the application interface is shown in Figure 2.

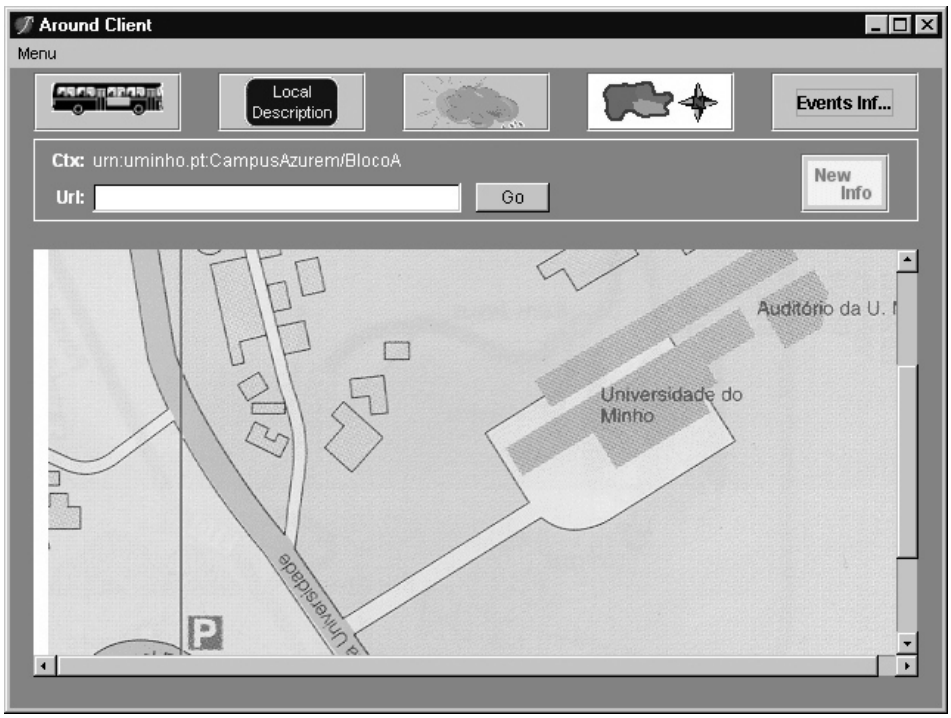

Figure 2: Screen shot of the application.

The interface is one of the most important parts of any application. In a case like this, where the user is possibly a tourist visiting an unknown place, we want to keep his attention on data and not in the application itself. Being simple is important, avoiding having users worried about how to use the application. Developing a specific interface for a kind of service and then extend that interface is a way that could be used by other services or develop a specific interface for each kind of service would probably result in a complex solution. Our implementation is based on a HTML capable area, implemented using the ICE Browser developed by ICEsoft[11]. This major area is used to display the data retrieved from the different services. The application was structured as a set of separated modules as depicted in Figure 3.

The Controller is the main module of the application, responsible for the initialisation of all the other modules. It also provides communication between the different modules.

The agents are autonomous units with all the knowledge necessary to use the services required for its thematic area. Each agent, knowing is current location context, must be able to contact the LBS server, find the services registered in that location context and then access and use the services. The agents encapsulate all the necessary knowledge to find and use the services, from query policies to the use the service methods. The display area of the application is an HTML area so each agent is also responsible for producing the HMTL file with the output to be shown to the users. An agent may be displaying information or may be in background when the display is at that moment assigned to another agent. While in background the agent may find new information to show to the users, because the user has changed to a new location context, the information itself changed, or just because a new service was, meanwhile, registered to that location. The new information found is signalled to the user by flashing the agent button on the top of the screen.

If one agent finds newer information than that being presented to the user it cannot just update the display. Suppose that we are walking on the street reading the history of the town castle that we have just visited when we enter into a new location context. Just changing the display wouldn't allow the user finish is reading. To avoid this problem the button placed at a navigation bar area is used to inform the user that new information is available.

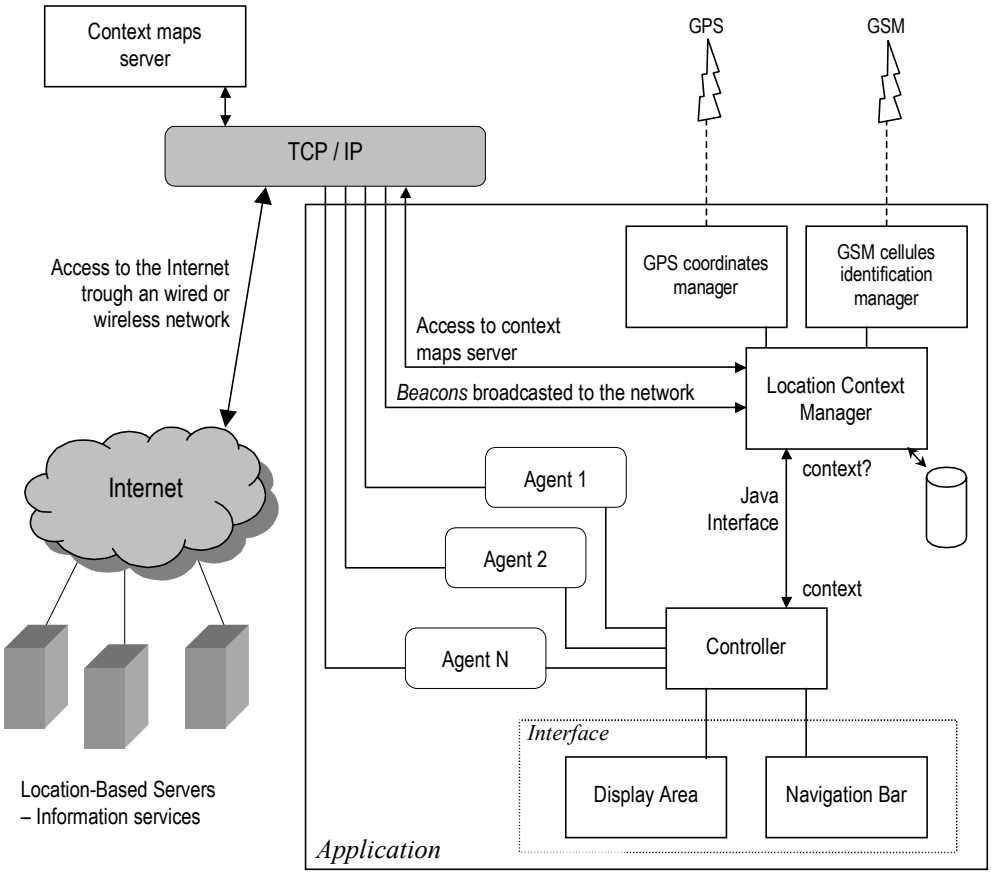

Figure 3: The application architecture.

The agents are also responsible for the implementation of a cache and a pre-fetching system. When entering a new location context the agents search for the services and immediately download the data to the users device if they found new information. If they find the same services that have been accessed recently, they just use the cached data, saving bandwidth. The use of pre-fetching allows the system to have the information immediately available when the user requests the information (change from one agent to another).

The Navigation Bar is a common area used to display the current location context and to allow users to write URLs and use this application as an ordinary browser. A simple button can be used by the agent currently using the display to inform the user that it has new information. By pressing the button the user indicates his intension to refresh the display, seeing the new information obtained by that agent.

The Location Context Manager is responsible for the determination of the current context. Presently it is working based on beacons broadcast through the network. In a new version under development the system will use GPS and possibly the cell ID of the GSM networks to achieve the current location. These last two technologies imply the use of a Contexts Map Server to convert the GPS coordinates or the GSM cell id into location contexts.

5. CONCLUSIONS

More than an application to assist a tourist, we have built a generic application for a set of location-based services. The modular construction of the application has the major advantage of allowing, with a small effort, the addition of new agents to access new services.

The construction of this application was used to test a new architecture for location-based services and provided a deep knowledge about how to deal with the new challenges that this kind of application raises, from the service selection to the interface. As future work we hope to study the possibility to dynamically download and use new agents to allow the building of a real generic application client of location-based services that could automatically become client of new services. To place an application client of location-based services inside a bus is also in our plans. This new application will allow the people travelling in the bus to have access to contextualized information, including information about the public transportation system of Guimarães town.

ACKNOWLEDGMENTS

The project “AROUND – Supporting Location-based Internet Services” was founded by the Portuguese Foundation for Science and Technology (FCT) (grant number PRAXIS/P/EEI/14267/1998).

Filipe Meneses master studies was founded by FCT (grant number PRAXIS XXI/BM/20895/99).

REFERENCES

1. R. José and N. Davies, "Scalable and Flexible Location-Based Services for Ubiquitous Information Access", *Lecture notes in computer science*, vol. 1707, Springer, Germany, 1999.

2. Y. Sumi, T. Etani, and et al., "C-MAP: Building context-aware mobile assistant for exibition tours", *Community Computing and Support Systems: social interaction in networked communities*, ed. T. Ishida, vol. 1519, pp. 137-154, Springer, Berlin, 1998.

3. Ollivetti, Active Badge web page, http://www.active-badge.com, 2000.

4. D. G. Abowd, G. C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: A mobile context-aware tour guide." Wireless Networks **3** , pp. 421-433, 1997.

5. K. Cheverst, N. Davis, K. Mitchell, A. Friday and C. Efstratiou, "Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences", Proceedings of Computer Human Interaction 2000, pp. 17-24, Netherlands, 2000.

6. HP Labs, Cooltown Web page, http://www.cooltown.hpl.hp.com, 2001.

7. Uwe Kubach and Kurt Rothermel, “Exploiting Location Information for Infostation-Based Hoarding”, in *Proceedings of The Seventh Annual International Conference on Mobile Computing and Networking*, July 16-21, pp. 15-27, ACM, Rome, Italy, 2001

8. L. Barfield, *The User Interface: Concepts & Design*, p. 353, Addison-Wesley, 1993.

9. P. Lettieri and M. B. Srivastava, "Advances in Wireless Terminals", *IEEE Personal Communications* **6**, pp. 6-19, 1999.

10. R. José, A. Moreira, F. Meneses and G. Coulson, "An Open Architecture for Developing Mobile Location-Based Applications over the Internet", *Proceedings of 6th IEEE Symposium on Computers and Communications*, Hammamet, Tunisia, July 2001.

11. IceSoft, IceSoft web page, http://www.icesoft.com, 2000.