A Mobile Agent Framework for M-Commerce

Patrik Mihailescu[1] Walter Binder[2]

[1] Peninsula School of Network Computing, Monash University, Australia, patrik77@optushome.com.au
[2] CoCo Software Engineering GmbH, Austria, w.binder@coco.co.at

Abstract

Opportunities in the development of new wireless m-commerce applications are enormous. Market research has valued these opportunities anywhere between 26 billion in 2004 to 42 billion in 2005. Within this paper we present an m-commerce framework based on agent technology. We believe that agent technology can greatly assist in the development of new wireless m-commerce applications as currently mobile devices suffer from limited networks bandwidth and processing power. A small testbed has been developed to demonstrate the potential of this framework.

Keywords: *M-commerce, mobile agents, wireless networks*

1 Introduction

It has been predicted that within the next few years the number of mobile devices connected to the Internet, such as cell phones, personal digital assistants (PDAs), pagers, and smart phones, will exceed one billion [1]. This potential growth has the ability to create new markets and opportunities in the development of wireless applications. Market research has valued these opportunities anywhere between 26 billion in 2004 [5] and 42 billion in 2005 [15]. We believe that there is a current lack of frameworks dealing specifically with the development of wireless applications. This may force companies into re-using existing frameworks that were designed primarily for developing Internet-based applications for personal computers. However, there are fundamental differences between developing wireless applications and Internet-based applications. For instance, wireless applications communicate over networks that suffer from low bandwidth, high error rates, and frequent disconnections when compared to wireline networks. Within this paper we propose the development of a new m-commerce framework based on the use of mobile agents. Through mobile agents our framework is able to overcome some of the limitations facing current generations of mobile devices (e.g., bandwidth limitations, low resources, etc.).

2 Current Frameworks for Development of Wireless Applications

Currently, wireless network providers are offering their services to customers through the Wireless Application Protocol (WAP) [14]. This allows customers to browse WAP-enabled web sites via a micro-browser installed within their mobile device. However, there are several problems associated with this:

- Currently, many mobile devices have limited screens, e.g., 160x120 pixels. This means they are only able to display small fractions of a web page at a time. Therefore, users have to spend considerable effort in scrolling before they are able to gather the information they are interested in.
- Due to bandwidth limitations, accessing web pages from a mobile device is significantly slower than access from a wired device.
- The user interface for WAP-enabled web sites is primitive when compared to traditional web sites. Devices that might be capable of displaying richer graphical user interface (GUI) screens through higher screen resolutions are not being exploited.
- Mobile devices communicate over wireless networks that tend to suffer from greater occurrences of disconnections due to a variety of problems, such as limited cell coverage, loss of signal, etc. Current frameworks only provide partial support for disconnected operation.

Despite these serious limitations, manufacturers of mobile devices as well as wireless network providers are still developing and deploying applications for mobile devices based on the same technology as traditional web applications. Instead of re-using existing frameworks, which will not work effectively for mobile devices, new frameworks specifically targeted at mobile devices are needed.

One approach to developing m-commerce frameworks proposed in [13] involves different companies developing specific layers that may be re-used by others. For instance, an organization specialized in wireless networks may develop a layer focussing purely on the lower level aspects of wireless network functions. This layer could be used by another organization that wishes to develop a wireless middleware product. Thus, the company developing the middleware product does not need to worry about the lower level details of the wireless network. Instead, they rely on the layer provided to them by another organization. This approach could be used within our framework, too, as network service providers may handle the specific details of how agents interface with other entities (e.g., agents, network services, etc.), while application providers develop wireless applications that use these lower level services.

3 Overview of a New M-Commerce Framework

Our m-commerce framework has been specifically designed to aid in the development of new wireless applications and services for mobile devices. Our approach differs from other frameworks that rely on the use of WAP technology in that we utilize mobile agent technology. The benefits gained from the use of mobile agent technology fall within three sections:

1. **Resources**: This can be broken down into two specific areas: network resources and computational resources. Through the use of mobile agents network resources are saved as an agent may be transferred to remote network locations, where it can perform the majority of a given task on behalf of a user. Furthermore this approach also saves computational resources of a mobile device, as processing is performed elsewhere within the network.

2. **Personalization**: Wireless applications developed using agent technology can be highly customized to individual users and their mobile devices. For instance, if a mobile device supports a higher screen resolution, an agent may take this into account by dynamically formatting the appearance of content it has collected from various network locations. Similarly, content could also be customized to an individual's preferences. For example, if an agent observes that a user only reads the sports section of an electronic newspaper and discards the rest (e.g., world news), it may download only the sports section in the future.

3. **New services**: Agent technology may assist in the development of next generation wireless applications and services. For instance, a shopping center could offer a web portal where consumers could send their agents from their cell phone to locate product specials that match their interests. Within the portal consumers' agents would negotiate with agents representing various stores, and deals could be reached between both parties. Once a deal is made, the agent could inform its owner with the specific details, such as which product was located, the terms and conditions of the deal, including directions on how to locate the store from their current position.

Our framework provides three types of agents:

1. *Device agent*: Each user is given a single device agent that will reside on the mobile device (e.g., cell phone or PDA). The device agent's responsibility is to enable a user to locate and to access various wireless services. This also includes handling certain tasks, such as negotiating for a service, payment of a service, etc.

2. *Service agent*: Service providers will use service agents to handle service requests generated by users. A service agent is considered a heavy-weight agent, as it offers a high level of functionality and only operates within the wired network.

3. *Courier agent*: Service agents do not communicate directly with a user, instead they use one or more courier agents for this purpose. Couriers are lightweight agents that only contain limited functionality and data applicable to the current interaction between a service agent and a user. Once a courier agent is transferred from a service agent to a device agent, it only communicates back with the service agent via a XML-based communication protocol. A courier agent cannot travel back to the service agent, to another device, or to another network location.

The benefits of using three different types of agents is that each one can be customized to a specific role. For instance, a device agent not only assists but also protects a user (from malicious courier agents) who is accessing various services. Both the service and courier agent play different roles and operate within vastly different environments – the wireless network and the wired network.

Wireless networks typically have lower bandwidth available than wired networks. This places a constraint on the size of a mobile agent when travelling through a wireless network. Therefore, we decided to create a smaller, lighter courier agent, one that only contains limited application logic. This not only means that courier agents can be transferred reasonable well through a wireless network, but they also help to reduce the amount of resources used when run on a mobile device due to their limited functionality.

Within the following subsections we will elaborate on the three types of agents.

3.1 Device Agent

Before any wireless services can be initiated, a user must firstly download an application that contains a device agent onto the mobile device. A device agent is a stationary agent that resides on a user's device (although it is possible to transfer this agent to another device) and provides access to various wireless services, such as location-based product comparisons. All services are accessed by a device agent via a pre-defined XML communication protocol.

Once a service has been agreed upon, all subsequent communications are between a device agent and a service agent. Communication between these two agents can be classed into two types: *system-level* and *user-level* communications. System-level communication is aimed at a device agent (not at a user) and is based on a XML communication protocol. An example of a system-level communication message is a task request. User-level communication is aimed at a user and is based on courier agents. An example of a user-level communication message is the presentation of the outcome of a service.

The device agent comprises of four layers, shown in figure 1. Each layer provides specific functionality:

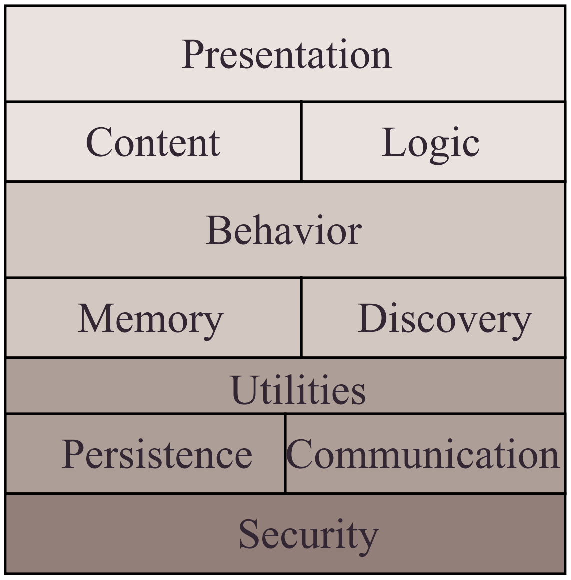


Figure 1: Structure of a device agent.

1. **Presentation**: This layer is responsible for managing the presentation of a wireless application, which will be contained within a courier agent. This layer provides two specific services to courier agents: *content management* and *logic control*. Content management deals with how information is displayed on a screen taking into consideration a user's personal preferences and the functionality of the device. Logic control deals with the interaction between information represented on a screen and the user (e.g., validation of a form).

2. **Behavior**: This layer is responsible for the limited intelligence of the device agent, which comprises of two parts: 1) Memory and 2) Discovery. As a device agent observers a users interaction with various services it may choose to store specific details about these interactions. This information can be used at a latter stage by the device agent when it wishes to handle some of the tedious tasks usually left to its owner. For instance if the device agent observes that on a daily basis its owner accesses the weather information for a specific county, it may automatically fill in this information instead of the user having to manually key in the county name.

3. **Utilities**: This layer is responsible for two specific services: *persistence* and *communication*. These services can also be requested by a courier agent. Persistence allows the device agent to store content in secondary storage that will remain once the device is switched off. For instance, a user may be half way through a transaction, when the network terminates. The device agent may choose to store some of the downloaded courier agents and their state in secondary storage so that the user could resume from their last point in the transaction once connected to the network. Communication provides a generic network protocol for the device agent to communicate with a service agent. For example, a device agent may use a HTTP or TCP socket connection when communicating with a service agent.

4. **Security**: This layer is responsible for securing any of the upper layers. It secures the communication

between the device agent and service agents. Furthermore, the security layer has to protect the device from malicious or badly programmed courier agents. It controls and limits the resource consumption of couriers (e.g., communication bandwidth, memory, CPU, etc.) and supports the revocation of resources and the safe termination of courier agents.

3.2 Service Agents

Service providers who offer consumers (connecting via wireless devices) access to their services also utilize agent technology. Service agents will be used by service providers to handle individual requests issued by users. As in other e-commerce frameworks based on mobile agents, such as e.g. in [2], service agents may migrate in the wired network in order to contact required services locally. For instance, service agents may be used for a distributed search in the World-Wide-Web.

The primary role of a service agent is to represent a user session within a service. In our framework a service agent is responsible for only a single user, therefore as more users access the service, additional service agents will be created to handle their sessions. The length of a session will vary from service to service. To personalize a service to individual users, service agents may store various pieces of information regarding an interaction (e.g., what request they submitted). This information could be used to offer specials or additional services to the user in the future.

To interact with users, service agents may transmit courier agents to the users' devices. A service agents may dynamically create a new courier, or it may pickup a predefined courier agent from a repository. Courier agents maintained within a repository contain generic application presentation logic, such as service selection or payment screens.

3.3 Courier Agents

Courier agents are single-hop agents that are transmitted to mobile devices from a service agent. A courier agent is comprised of two parts:

- the information to be displayed to a user (agent state), and
- the business logic that determines how that information is manipulated (agent code).

Courier agents typically have a short life span, once they have displayed their information they may be destroyed or cached depending upon the application. Courier agents rely on the services offered by device agents. For instance, a courier agent does not directly display its information to a user, but makes a call to the presentation service of a device agent to render its contents to the screen. Using this approach, courier agents are independent of the underlying device. Moreover, the device agent is able to mediate all actions of a courier. All communication between a courier agent and the service agent is based on XML messages.

In complex applications a single courier agent does not need to encompass the entire user interface, instead it may only be responsible for a small subset. Service agents are in charge of supplementing new courier agents as required. This structure helps to minimize network communication, as once downloaded courier agents may handle a series of user dialogs before a new courier agent is needed. This approach offers the following two benefits:

1. If a user disconnects from the network, he will still be able to interact with the courier agent off-line. Once on-line, the courier agent may resume its session with the service agent. The acceptable length of inactivity will differ from application to application. For instance, a conference reviewing system may permit an infinite period of inactivity, whereas an on-line store may only tolerate an hour of inactivity before terminating the session.
2. Mobile devices suffering from limited resources will benefit from this approach, as only a small subset of the presentation logic of the application will be loaded. This may be further optimized by service agents using adaptive algorithms to fine tune the functional range of courier agents to suite the behavior of certain users and the computational power of their mobile devices.

Figure 2 presents a sample scenario using our framework. The steps that occur within this scenario are as follows:

1. A user locates and issues a request to a service provider.
2. A service provider creates a service agent to handle the user's service request.
3. The service agent and the device agent begin negotiations for the service.
4. The service agent moves to another network location to fulfill the user's request.
5. The service agent communicates with the user by sending courier agents.

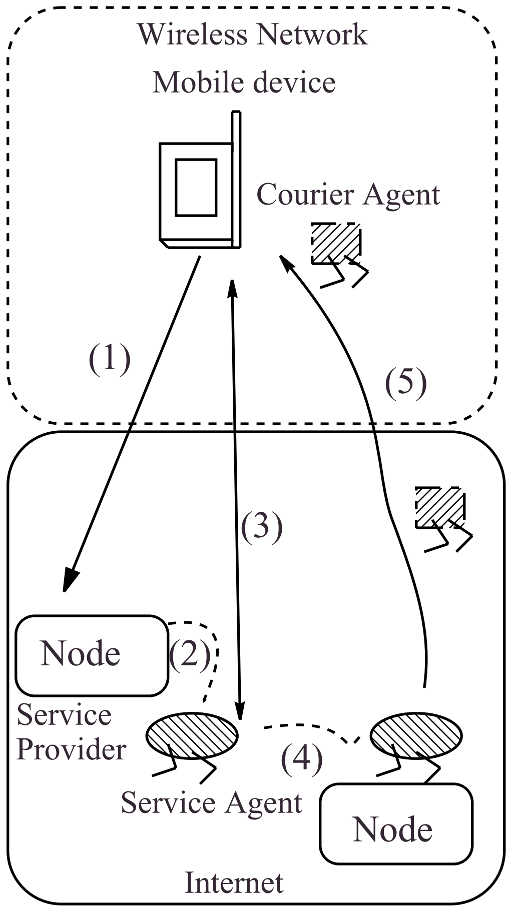


Figure 2: Overview of our m-commerce framework.

Within our framework there are restrictions on the interaction that can occur amongst the three different types of agents. These restrictions are presented within table 1. The reason for these restrictions is that we wish to isolate different applications from each other, to protect them from unwanted side-effects. The major restrictions are placed upon service agents and courier agents. A service agent is only able to communicate with a device agent that initiated the service and only with courier agents that it has generated. The same principle applies for courier agents, they are only able to communicate with the service agent they originated from and other courier agents from the same service agent.

- O: Unrestricted communication.
- X: Restricted communication to a single device agent that initated a service.

Table 1: Communication restrictions.

	Device A.	*Service A.*	*Courier A.*
Device A.	O	O	O
Service A.	X	O	Ox
Courier A.	X	X	Ox

- Ox: Restricted communication to a group of courier agents originating for the same service agent.

4 Testbed

A small testbed has been developed to test the viability of our framework. The testbed is based on a shopping center scenario, where consumers can access a web portal wirelessly via their PDA device for services such as:

1. **Product locator**: A service where a specific product can be located within the shopping center. This service returns a list of possible stores that stock the request product, including pricing information and the location of the store.
2. **Product comparison**: A comparison service that locates a specific product based on given criteria. For instance, a user may only wish to purchase the product if a two year warranty is given, while another user may only purchase the product if it is within a certain price range.
3. **Store locator**: A service, which locates a specific store within the shopping center. This service returns directions on how to locate the particular store.
4. **Specials locator**: A service that actively seeks out product specials offered by stores within the shopping center. Users can choose the types of product specials they are interested in, e.g., electronics, clothes, shoes, sunglasses, etc. The service returns an electronic coupon that users can redeem at a store.

Figure 3 gives an overview of the testbed scenario. A web portal acts as a central location where users can access the four listed services via a wireless PDA. Service agents will reside within this web portal. Some of the larger stores within the shopping center may have a network host, which service agents could travel to while fulfilling a given request. Other smaller stores that do not have a network host could store some of their product information in the web portal itself. Within the network hosts, other agents (representing the interests of the stores) could negotiate with incoming service agents.

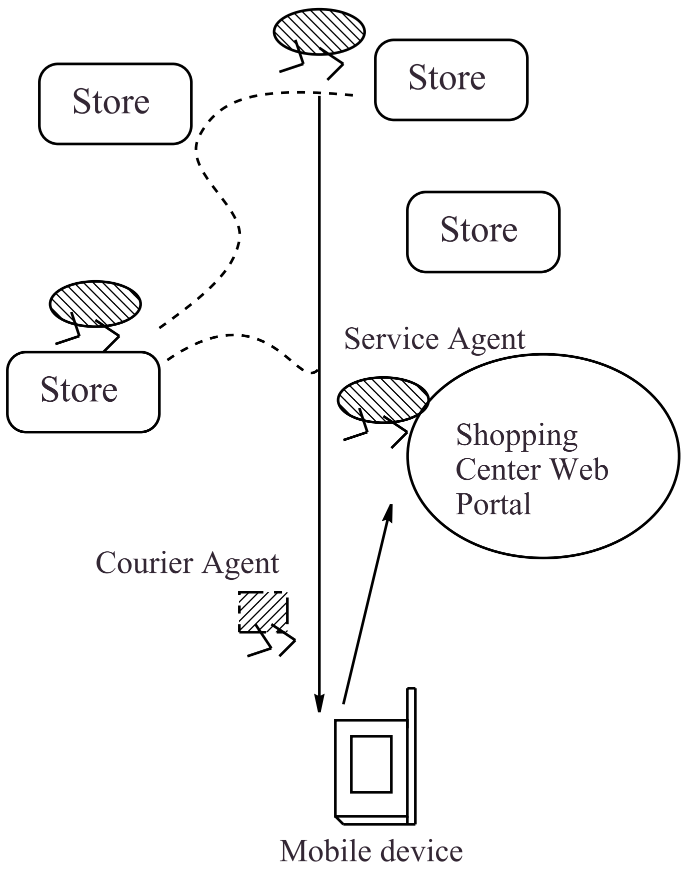


Figure 3: Overview of our testbed.

Currently, only the specials locator service is being implemented. We will now discuss the implementation details of the testbed.

4.1 Java 2 Micro Edition (J2ME)

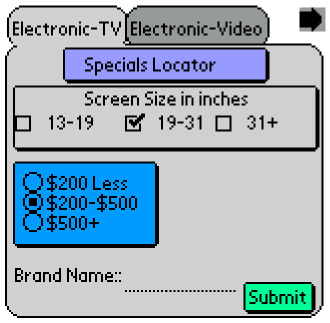
The J2ME is aimed at developing Java applications for two types of devices: fixed or mobile. Fixed devices operate with greater resources (e.g., 2–16 MB of memory) and communicate via a fixed wireline network. Devices such as web phones and TV boxes fall within this classification. Mobile devices operate with substantially lower resources, such as 120–500 KB of memory, and communicate via a wireless network. Cell phones, PDAs, smart phones, pagers, and household appliances fall within this classification. To distinguish between both devices, J2ME defines two separate specifications: *Connected Device Configuration (CDC) [7]* for fixed devices and *Connected Limited Device Configuration (CLDC) [6]* for mobile devices. We are using the CLDC specifications for our testbed work.

Within the CLDC specification there is a concept known as *profile*. A profile defines a common set of APIs that are applicable for a certain product range (e.g., cell phones). It will be up to manufacturers of different devices to come up with their own profile as reviewed within the Java Community Process (JCP). The major benefit for multiple profiles is that each profile will only contain essential API sets. This is extremely important as the intended devices suffer from limited resources and any attempt to overburden them with additional APIs will only slow them down. Currently, there is only one official profile available, aimed at cell phones. It is called the Mobile Information Device Profile (MIDP). There is a second profile currently undergoing review, aimed at PDA devices. As a temporary measure Sun has released a development kit called the KVM [12]. This currently runs on the Palm operating system, Windows, and Linux. It is the development kit we used for our testbed.

4.2 Testbed Implementation

All three types of agents within the testbed were implemented using Java-based tools. The service agents were developed using the Aglets SDK[9], while the device agent and courier agents were developed using the KVM SDK. The implementation for both the device agent and courier agents is based on an existing platform called MAE [11]. In the following we provide some implementation details of the layers for a device agent:

1. **Presentation – Content**: The type of user interface objects that could be used by courier agents are: button, radio button, text field, check box, tabbed pane, etc.
2. **Presentation – Logic**: A call-back routine similar to the standard Java event handling model was used.
3. **Behavior – Memory**: Only the type of service that was accessed by a user was stored by a device agent. The information was stored within a Palm database. The database size was only limited by available storage space on the PDA.
4. **Behavior – Learning**: This service was not implemented.
5. **Utilities – Persistence**: Courier agents could be stored within a Palm database. The size was limited to the available storage of the PDA.
6. **Utilities – Communication**: There were two communication mechanisms supported, TCP sockets and IrDA.

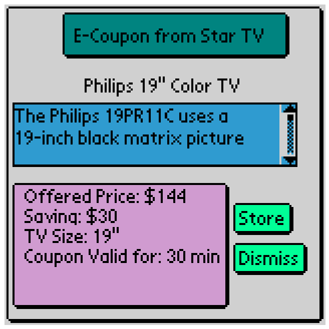


Screen Size in inches
13-19 [✓ 19-31] 31+

$200 Less
[✓ $200-$500]
$500+

Brand Name::

Figure 4: Courier agent displaying the service options.



Philips 19" Color TV

The Philips 19PR11C uses a 19-inch black matrix picture

Offered Price: $144
Saving: $30
TV Size: 19"
Coupon Valid for: 30 min

Figure 5: Courier agent presenting the results from the service.

7. **Security**: This layer was not implemented, but there are several third-party encryption packages available, such as the Bouncy Castle Cryptography Package [10], which could be used.

The testbed was tested on a small LAN network, consisting of four desktop computers, a single wireless access point, a network hub and a wireless PDA device. The PDA device was a Handspring Visor that contained a module from Xircom enabling the PDA to access a 802.11b Wireless LAN. Two screenshots from the service running on the PDA are shown in figure 4 and 5.

The limitations within this testbed are that courier agents need to specify in actual pixels, the location of their GUI objects. Currently the device agent was not able to provide a mechanism that would enable courier agents to specify a weighting on which GUI objects get greater preference. Furthermore the KVM is at least three times slower than native C, which effects the transmission speeds of courier agents and general performance on the PDA.

5 Conclusion

In this paper we have presented a m-commerce framework based on mobile agent technology. This framework offers the following benefits:

1. Through the use of mobile agents, we are able to offload the majority of the processing onto servers without the need of constant network communication.

2. Only a subset of an application is loaded onto a mobile device, which assists in minimizing resource usage.

3. Off-line processing is supported, as well as the ability to resume processing in case of disconnections.

Future areas that need to further explored within our framework include:

- Develop a relative GUI placement mechanism within the device agent, which will allow courier agents to display their information regardless of the screen dimensions of the underlying device.
- Develop a true push mechanism that will enable alerts to be sent from a service agent to a device agent, rather than the device agent having to poll the service agent if any courier agents are to be sent.
- Investiage the real feasibility of a courier agent operating in offline mode. What sort of resources are needed at both locations to keep the service agent and courier agent in storage.
- Examine other development kits that could be used to implement this framework, such as J9[4], JBed[3], Kada[8], etc. Are there any performance improvements over the KVM development kit.

Acknowledgements

The authors would like to thank Jarle Hulaas and Klaus Rapf for their valuable support.

References

[1] AllNetDevices. Wireless data access to outpace wired. Web pages at `http:`

//www.allnetdevices.com/industry/market/2000/10/26/wireless_data.html, Oct. 2000.

[2] P. Dasgupta, N. Narasimhan, L. E. Moser, and P. M. Melliar-Smith. MAgNET: Mobile agents for networked electronic trading. *IEEE Transactions on Knowledge and Data Engineering*, 11(4):509–525, 1999.

[3] Esmertec. jbed product line: Whitepaper. Web pages at http://www.jbed.com/p_whitepaper.html.

[4] IBM. Java development for embedded systems. Web pages at http://www.embedded.oti.com/learn/vaesvm.html.

[5] IDC. Smart handheld devices will represent a $26 billion opportunity by 2004. Web pages at http://www.idc.com/Hardware/press/PR/PS/PS022601pr.stm, Feb. 2001.

[6] Java Community Process. Jsr-000030 j2me connected, limited device configuration. Web pages at http://jcp.org/aboutJava/communityprocess/final/jsr030/index.html.

[7] Java Community Process. Jsr-000036 j2metm connected device configuration. Web pages at http://jcp.org/aboutJava/communityprocess/final/jsr036/index.html.

[8] Kada Systems. Kada vm. Web pages at http://www.kadasystems.com/kada_vm.html.

[9] D. B. Lange and M. Oshima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1998.

[10] Legion of the Bouncy Castle. Home page. Web pages at http://www.bouncycastle.org/.

[11] P. Mihailescu and E. A. Kendall. Development of an agent platform for mobile devices using j2me. In *In Proceedings of the Evolve 2001 conference*, Sydney, Australia, May 2001.

[12] Sun Microsystems. Java 2 platform micro edition (j2me) technology for creating mobile devices. Web pages at http://java.sun.com/products/cldc/wp/KVMwp.pdf.

[13] U. Varshney and R. Vetter. A framework for the emerging mobile commerce applications. In *34th Hawaii International Conference On System Sciences (HICSS-34)*, Maui, Hawaii, Jan. 2001.

[14] WAP Forum. What is WAP? Web pages at http://www.wapforum.org/what/technical.htm.

[15] Wireless Today. Wireless today: Market snapshot. Web pages at http://www.wirelesstoday.com/snaparchives/snap070300.html, July 2000.