An Enterprise Messaging Middleware for Mobile Services

Brahim Medjahed
Department of Computer and Information Science
University of Michigan, Dearborn
brahim@umich.edu

David Carver
School of Engineering
University of Michigan, Ann Arbor
carverd@umich.edu

Abstract

In this paper, we introduce a messaging middleware for enterprise services in wireless environments. The proposed middleware enables seamless integration between mobile devices/users and enterprise services. Mobile users send their requests to the enterprise system which replies by sending back the requests' results to the mobile user. We adopt object-oriented analysis methods to provide for in-depth description and analysis of the messaging middleware.

1. Introduction

The concept of Web services is emerging as the technology of choice for enabling intra- and inter-enterprise collaborations [ACK03]. Simply put, a Web service is a set of related business functions that can be programmatically accessed through the Web [ACK03]. The widespread adoption of Web services has spurred an intense activity in research and industry to address various Web service issues. However, most of the existing techniques were developed for wired infrastructures with fixed or stationary users [MM06]. The past years have witnessed a boom in wireless technologies and devices [YBM03]. To support wireless-oriented services, a new generation of Web services called mobile services (m-services) has emerged. An *m-service* is a Web service that is accessible by mobile hosts through wireless networks.

We identify two business models for mobile services [LDV05]: *business-to-consumer* and *business-to-employee*. In the business-to-consumer model, customers use mobile devices to access and pay for enterprise services. In the business-to-employee model, mobile devices are used to integrate employees, which are traveling (e.g., sales and maintenance people), into the daily business of the enterprise. Our focus in this paper is on enabling access to m-services in the business-to-employee model.

To illustrate the advantages of providing access to enterprise services via mobile devices, let us consider the following scenario. A hospital messaging system enables physicians (mobile users) to access services via their PDAs (Personal Digital Assistants). Physicians make service requests and receive responses through the hospital messaging system. Examples of services include a prescription compatibility service (e.g., checking that the prescriptions currently taken by the patient are compatible with the suggested treatment), ambulance alerts (e.g., an alert stating the physician is needed in the emergency), and access to the patient's record (e.g., checking the patient's history).

As stated in the previous example, one key component for enabling access to enterprise mobile services is the *enterprise messaging middleware* (EMM). Numerous applications are required to interact with each other in support of an enterprise [Yua04]. The EMM facilitates such interactions; it provides a common, reliable way for applications to create, send, receive, and read messages. A well-adopted messaging model is currently defined for Web services as witnessed by the recent standardization efforts (see the reference model for Web service in [ACK03]). However, there is currently no agreement on the definition of a messaging middleware for mobile services. Herein, we propose an EMM for accessing enterprise services from mobile devices. We present an overview of the EMM model in Section 2. We give an inside view of the EMM in Section 3. Section 4 discusses implementation issues. Section 5 is devoted to related work. We present concluding remarks and future work in Section 6.

2. The Enterprise Messaging Model

In the section, we describe the basic components of the proposed middleware. First, we define the different types of EMM users and services. Then, we illustrate the way users and services connect to the EMM.

2.1. Mobile and Administrative Users

There are two types of users in the system: *mobile users* and *administrative users*. *Mobile users* (e.g., physicians, nurses) access enterprise services via a PDA. They are responsible for gathering data and entering it into the mobile device. We categorize use cases for mobile users into two categories:

27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07)
0-7695-2838-4/07 $20.00 © 2007 IEEE

- *Data collection*: mobile users enter information (e.g., heart rate, blood pressure, and physical condition) into a mobile device.
- *Service Invocation*: mobile users invoke enterprise services (e.g., diagnosis service) via their mobile devices.

Administrative users (e.g., system administrators) are responsible for clustering devices into *groups*. *Groups* are useful for deploying software (e.g., transcription) to mobile devices and determining the users of a mobile device. For example, users in a cardiac unit will have different software needs than users in the maternity ward. Administrative users must be clear on the different needs of each group and leverage requirements as part of the device group. The enterprise system maintains the list of software on each group and maintains the latest version of the software. It ensures that the right software is applied to the right mobile device. Using groups to manage mobile devices, allows administrators to add a new device without having to define the functionality of one device. The following use cases apply to the administrative user:

- "Define Device Groups" allows administrative users to define groups of mobile devices. Each mobile device is associated to a particular group.
- "Define Software Groups" allows administrators to define software that is associated with a specific device group. For example, a cardiac care mobile user group may need to have particular software associated with its users.
- "Get Device Information" enables administrators to request information about a specific device. This information includes the user, its group, how long it has been used, and the software currently installed.
- "Deploy New Software" enables new software to be deployed to a device group. The enterprise system determines which devices need to updated and then deploys the software to the devices of that group.

2.2. Different Types of Services

Enterprise services are of two types (Figure 2): *business* and *administration*. *Business services* perform the daily activities of the enterprise (e.g., prescription compatibility service and diagnosis service). Users invoke business services via *software* (also called *mobile applications*) deployed in their mobile devices. For example, physicians verbally record patient data digitally using medical transcription software deployed on their PDAs.

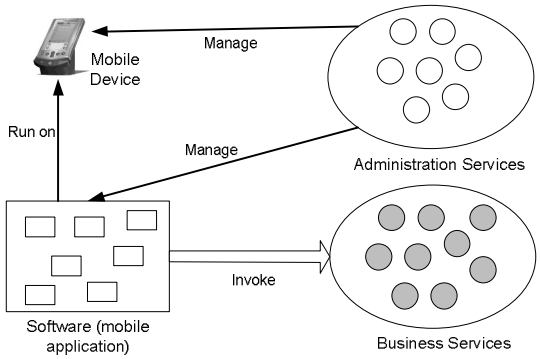

Figure 2. Different Types of Services

Administration services provide functionalities necessary to the execution of business services. One such service is a *configuration service*. This service instructs the mobile device on how it should be configured. It would inform the mobile device as to the parameters it should gather (e.g., body temperature, heart rate). After the mobile device has been turned on, it will send a message to a server, informing the server that it is needs to be told about how it should be configured. The server then acquires the mobile device's specific configuration information and returns a response containing configuration information. The mobile device will take the configuration information and configure itself appropriately

2.3. Connectivity

One important issue in the EMM is the way applications connect to the messaging provider. The EMM supports a *proxy* that plays the role of the mobile device in the enterprise. A *proxy* is a structural pattern that provides a local representation of an entity in a different address space [GHJ95]. The mobile device communicates with the enterprise messaging system through the proxy over a wireless connection (e.g., 802.11b connection). The proxy receives a mobile message from the mobile device, passes it to the messaging provider, which forwards it to the enterprise system's *service broker* as an enterprise message. The services broker provides a single point of reference for the proxy server. It looks up the desired service in a services repository (e.g., UDDI). The services broker implements the *façade pattern* which provides a unified interface to a set of interfaces in a subsystem. By using the façade pattern [GHJ95], the services broker

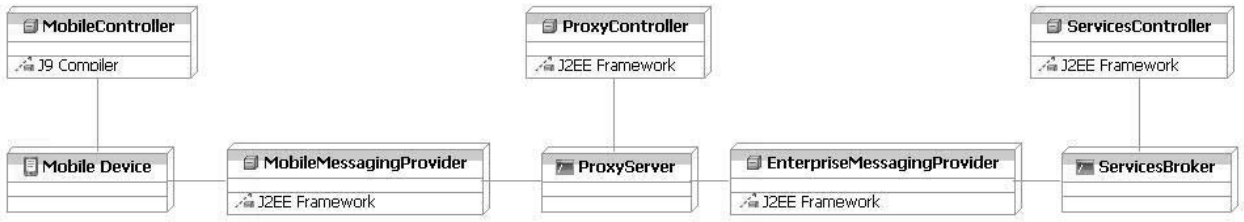

Figure 4. System Objects and Interactions

provides a standard way to access enterprise m-services. The services broker makes interactions with disparate services transparent to the proxy.

We identify three domains in the enterprise messaging architecture: the *wireless domain* containing the mobile device and the proxy server, the *messaging* domain containing the proxy server and the services broker, and the *enterprise domain* containing the services broker and the various services servers located in the enterprise network. The messaging provider is responsible for the delivery of (mobile) messages to the mobile device from the proxy server. It is also responsible for delivering (enterprise) messages between the proxy server and the enterprise services broker. The proxy server contains a queue that holds messages sent to and from the mobile device. Mobile messages are lightweight messages (e.g., kSOAP) compared to enterprise messages (e.g., SOAP).

One key aspect of the system is the ability to send and receive messages from the mobile device and proxy server seamlessly. The proxy server must be able to "detect" when the mobile device has come into a range where the two can communicate and then facilitate that communication. The proxy server and the services broker will be connected through a local area network (LAN) or wide area network (WAN). The mobile device and proxy server communicate by the messaging provider. The EMM will provide seamless integration between the mobile device and the proxy server. The proxy server will allow the mobile device to physically be in the wireless network and logically be in the enterprise network.

3. An Inside View of the EMM

In this section, we use object-oriented analysis for describing our middleware. First, we present the system objects and their interactions. Next, we introduce interface objects and explain how they are used to facilitate communication within the messaging system. Finally, we use sequence diagrams to explore the way messages are sent throughout EMM objects.

3.1. EMM Objects and Interactions

The messaging middleware objects can be decomposed into two areas (Figure 4): *messaging providers* and *controllers*. We identify two types of messaging providers: m*obile messaging provider* and e*nterprise messaging provider*. The difference between the mobile and enterprise messaging providers is on the type of messages they carry. The mobile messaging provider carries mobile messages. The enterprise messaging provider carries enterprise messages. We use OpenJMS as mobile and enterprise messaging providers.

Controllers are used to send and receive messages. Applications interact with controllers which interact with other controllers. We identify three types of controllers: *mobile controller*, *proxy controller*, and *service controller*. The type of messages (mobile or enterprise) sent/received by a certain controller depends on the context that the controller functions in. For example, the *mobile controller* functions in the mobile environment so it sends and receives mobile messages. The *proxy controller* sends and receives both mobile and enterprise messages. The s*ervices broker controller* sends and receives enterprise messages. Mobile applications interact with a mobile controller that communicates with a proxy controller, which interacts with the services broker controller.

3.2. Structural Views of Interface Objects

The class diagram that illustrates the structural view of the interface objects is presented in Figure 5. The interface objects use two structural patterns: *proxy* and the *façade patterns* [GHJ95]. The *proxy pattern* allows for communication of the mobile device and a proxy server to be decoupled from the environment. This way, proxies can be used to represent the mobile device in the wireless domain. A proxy represents any number of mobile devices in a given region. For example, in the scenario above, we may have a proxy for Intensive Care Unit patients, a proxy for mother of newborn patients, and so on. The "ApplicationController" and "ProxyController" use the

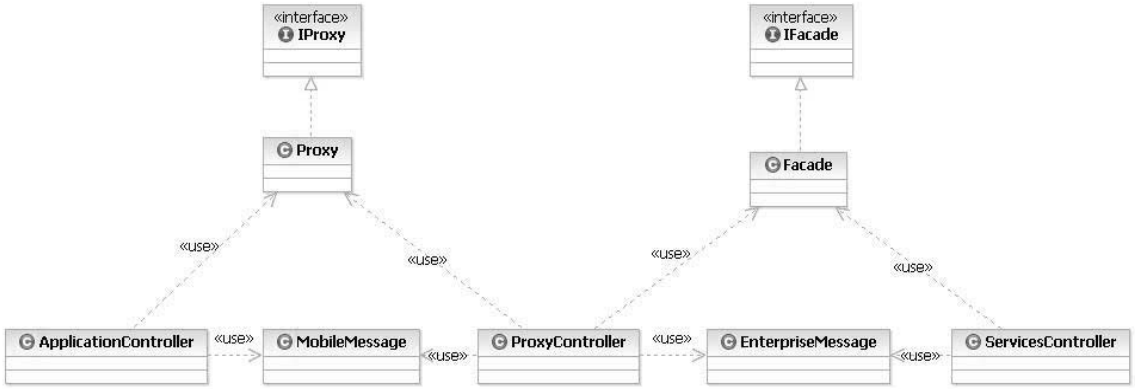

Figure 5. Structural View of Interface Objects

proxy pattern.

The *façade pattern* facilitates a centralized mechanism for gaining access to the services on an enterprise network. It provides a unified interface to a set of interfaces in a subsystem. The "ServicesController" and "ProxyController" classes reside on the services broker and proxy server, respectively. The services broker receives requests from the proxy server and then returns a response to the proxy server which then marshals the responses on to the mobile device. The "ServicesController" and "ProxyController" use the façade pattern. The "ServicesController" manages the interactions between the "ProxyController" and the other services provided by the enterprise. Applications that reside on the mobile device use the "ApplicationController" class to communicate with the enterprise. The "ApplicationController" sends and receives mobile messages from the "ProxyController".

3.3. Object Bahavior

Object behavior refers to the actions and reactions of object interactions in an enterprise system. We use sequence diagrams to express this behavior and model sequence logic. The key classes include (i) the "Controller" classes: "ApplicationController", "ProxyController", and the "ServicesController"; and (ii) "Messages" classes: "MobileMessage" and "EnterpriseMessage".

The sequence diagram in Figure 6 illustrates how a mobile device sends a message to the services controller. The mobile user creates a "MobileMessage" object through an application running on the mobile device. The application uses an "ApplicationController" object to pass the message to the "MobileMessagingProvider" object. Once the "ProxyController" receives a mobile message from the mobile device, it creates an "EnterpriseMessage" object and forwards it to the enterprise messaging provider. The "EnterpriseMessagingProvider" object transports the message to the "ServicesController" object.

Once the "ServicesController" receives a message from the "EnterpriseMessageProvider" object (Figure 7), it locates the service that has been requested, assembles a response, and sends it to the "ProxyController" as an "EnterpriseMessage". The "ProxyController" then creates a "MobileMessage" and transports it using the "MobileMessagingProvider" to the "ApplicationController".

4. Implementation

Figure 8 gives an overview of the deployment architecture of the messaging middleware. It includes deployment specifications such as J9 compiler, the J2EE framework, and APIs such as kSOAP and OpenJMS (which also serves as the messaging provider) [Wil03, Yua04].

On the mobile device, we use J9 compiler, IBM's implementation of Sun's Java Virtual Machine specification. The mobile device sends/receives requests/responses to/from the proxy server using kSOAP messages. The proxy server uses Apache Tomcat to provide a servlet container. It uses kSOAP package and the OpenJMS package as a messaging provider. It also uses other APIs that are part of the J2EE framework and the Apache Axis Soap implementation. Like the proxy server, the services broker uses J2EE framework and the Apache Axis implementation of Web services and SOAP. The services broker finds the appropriate service in the enterprise and makes the request and gets the response from it. It then assembles the response into an enterprise message and returns it to the proxy server. Let us now look at how a client would "get" a message from the proxy server. Assume that a

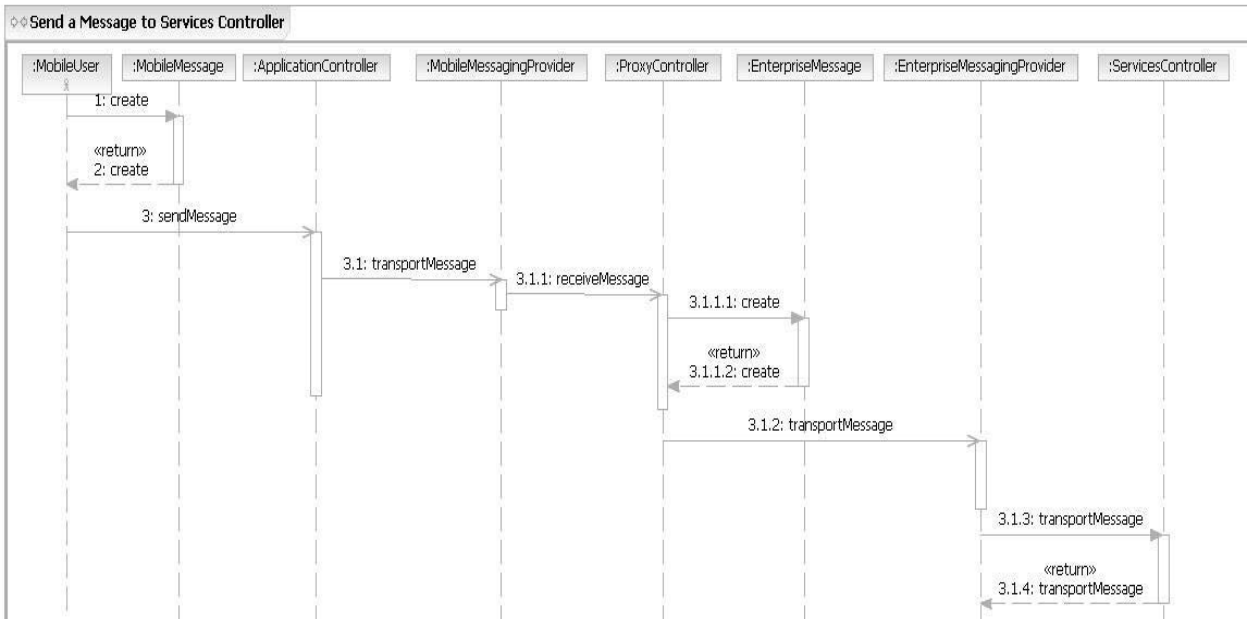

Figure 6: Send a Message from Mobile Device to Services Controller

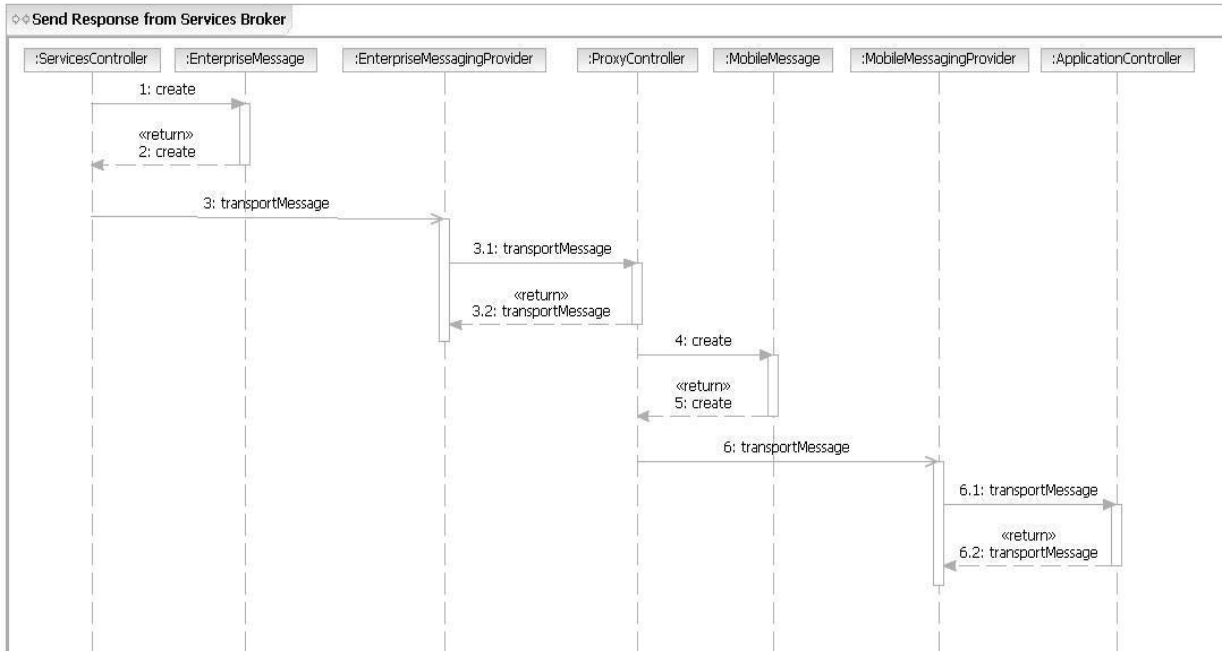

Figure 7: Service Broker Sends Response to Mobile Device

physician has made a request for a diagnosis suggestion. The request has traveled from the mobile device to the proxy server over a kSOAP message. The proxy server then adds the kSOAP message to the OpenJMS message queue. The kSOAP message is then packaged into an enterprise message and sent to the services broker's queue. The service broker then takes the message, finds the service on the network, gets the response, packages the response into a message, and sends it to the proxy server. The proxy server takes the response from the queue, un-packages it, and packages it as a kSOAP in a queue. After the aforementioned events have occurred, there is now a message to be consumed by the mobile device.

5. Related Work

[PTH02] introduces a proxy-based architecture for m-services. The proxy interacts with a services broker

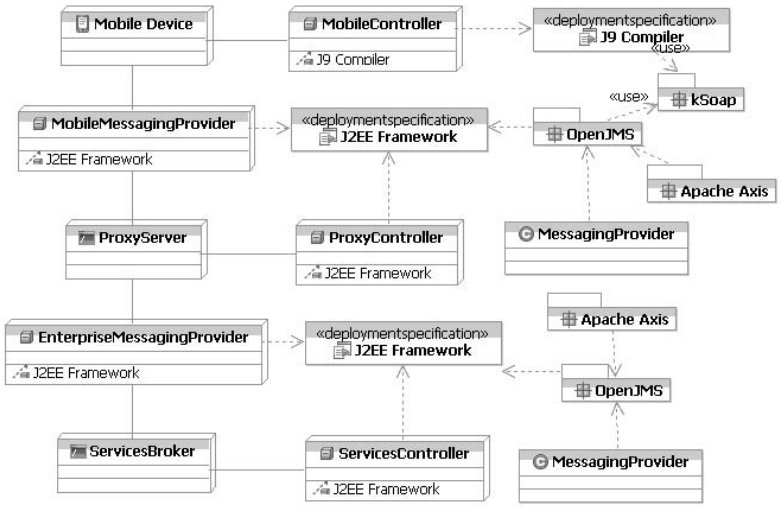

Figure 8. Implementation Architecture

and provider and returns the results to the mobile device using Web services agnostic protocols such as WAP/WML. In our solution, we use messaging providers between the mobile device and the proxy server, and the proxy server and services broker. We utilize the concept of messages throughout the enterprise and kSOAP for consumption and production of messages. [DJO02] addresses the issue of limited bandwidth. We solve this issue by defining groups of mobile devices. For example, a maternity ward group will contain any number of proxy servers that allow for high availability between the proxy server and its mobile device group. If connectivity becomes an issue, more proxy servers will be added for a specific group. [MYM01] presents an agent-based approach for invoking m-services. Mobile users submit their request to a user-agent located in the wired network. The user-agent will invoke the requested service on behalf of the user and send the results back to that user. [MYM01] does not focus on describing the messaging middleware between the mobile user and user-agent. Our paper presents an end-to-end middleware for executing users' requests in wireless environments.

6. Conclusion

In this paper, we propose an EMM for accessing m-services. The middleware uses two structural patterns to pass messages from mobile devices to enterprise services and back to mobile devices. Future work includes a performance study of the EMM. We will measure performance both at the mobile device (e.g., power consumption) and enterprise (e.g., response time under various loads). We are also planning to develop security mechanisms (e.g., access control) for the EMM.

References

[ACK03] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. "Web Services: Concepts, Architecture, and Applications". Springer Verlag, June 2003.

[DJO02] D. Dahlem, J.H, Jahnke, A. Onabajo, and O. Wang. "The Challenges of Implementing Web Services on Small Devices". OOPSLA Workshop Pervasive Computing, November 2002.

[GHJ95] E. Gamma, R. Helm, R. Johnson, J. Vlissides. "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, 1995.

[LDV05] P. Langendörfer, Z. Dyka, F. Vater, R. Kraemer "Integrating Mobile Devices into E-business Architectures: Open Issues and Potential Solutions", IT-Symposium der HP User Society, 2005.

[MM06] Q.H. Mahmoud, Z. Maamar. "Engineering Wireless Mobile Applications". International Journal of IT & Web Engineering, 1(1), January 2006.

[MYM01] Z. Maamar, H. Yahyaoui, W. Mansour, and W.-J. Vd Heuvel. "Software Agents and Wireless E-Commerce". ACM SIGecom, 2(3), June 2001.

[PTH02] T. Pilioura, A. Tsalgatidou, S. Hadjiefthymiades. "Scenarios of Using Web Services in M-Commerce", ACM SIGecom, 3(4), December 2002.

[Wil03] D. Wilding-McBride. "Java Development on PDAs: Building Applications for Pocket PC and Palm Devices" Addison-Wesley, 2003.

[YBM03] X. Yang, A. Bouguettaya, B. Medjahed, H. Long, W. He. "Organizing and Accessing Web Services on Air". IEEE Transactions on Systems, Man, and Cybernetics, 33(6), November 2003.

[Yua04] M. J. Yuan. "Enterprise J2ME: Developing Mobile Java Applications". Prentice-Hall, 2004.