A Framework for Over the Air Provider-initiated Software Deployment on Mobile Devices

Chuong Cong Vo
Department of Computer Science
& Computer Engineering,
La Trobe University, Bundoora, Melbourne,
Victoria 3086, Australia
Email: ccvo@students.latrobe.edu.au

Torab Torabi
Department of Computer Science
& Computer Engineering,
La Trobe University, Bundoora, Melbourne,
Victoria 3086, Australia
Email: t.torabi@latrobe.edu.au

***Abstract*— While mobile users want to obtain software anywhere and anytime without the knowledge of software availability in advance, software providers want to deploy their software to mobile devices without the users' intervention. However, because mobile devices expose special characteristics such as restriction of capability, heterogeneity, mobility, wireless connectivity, and context-orientation, deployment of software or context-aware services on mobile devices is a problematical issue. This work proposes a framework for over the air provider-initiated software deployment on mobile devices (OTA-PSD). Using the OTA-PSD, once new software or software updates are available, by the initiation of the provider, the compatible mobile devices, which have subscribed to the provider, will automatically receive this software with the minimum users' intervention. This framework allows providers to deploy their services on mobile devices based on users' context and device specification. To demonstrate the feasibility of the OTA-PSD, we developed a deployment system based on the OTA-PSD. We successfully demonstrated the deployment of a case study using our deployment system.**

***Index Terms*—Software deployment, Pervasive computing, Mobile devices, Software and Device specification**

I. Introduction

Mobile devices such as PDAs and mobile phones have been proliferating. The demand for software on mobile devices is more and more increasing. Moreover, software on mobile devices needs to be updated frequently because of frequent changes of users' context. Consequently, this pressing demand is expecting an effective solution for deployment of software and software updates on mobile devices. However, there is no solution to deploy software or software updates on mobile devices without users' discovery or knowledge of software availability in advance.

This work investigates existing approaches of over the air (OTA) software deployment on mobile devices. Consequently, we propose a framework for OTA provider-initiated software deployment on mobile devices (OTA-PSD). Our framework supports both push model and pull model for software deployment. It allows software providers to push their new software or software updates to mobile devices based on users' interest, context and device capability. In order to demonstrate the feasibility of the OTA-PSD, we developed a deployment system based on the OTA-PSD. We also present a case study to demonstrate the operation of this system.

The rest of the paper is organised as follows. Section II presents some background of software deployment on mobile devices. Section III investigates previous work in the field of software deployment. Section IV and V describe the design of the OTA-PSD and the implementation of the deployment system based on the OTA-PSD. Section VI demonstrates the deployment of an example application on mobile device. Section VII presents an evaluation of the deployment system to confirm the advantages of our solution. Finally, we conclude this research and give some directions for future work.

II. Background

In this section, we presents some concepts regarding software deployment. We also outline the specific requirements and features of software deployment on mobile devices. We refer to these requirements as criteria to evaluate an approach to software deployment on mobile devices.

A. Software deployment

Software deployment is referred to a collection of activities, which are to make software available for use, until uninstalling it from devices [2]. These activities include delivery, installation, configuration, activation, updating, reconfiguration, and un-installation of the software [3].

B. Requirements of software deployment [3], [4]

1) Minimum resource consumption: Any approaches to software deployment on mobile devices should take resource restrictions of mobile devices into account.
2) Generality: A deployment mechanism can be applied for various kinds of mobile devices with multiple platforms.
3) Adaptation: Software deployment must adapt to users' context and device capability.
4) Automation: Software deployment can be performed automatically without users' intervention.
5) Supporting continuous services: Software can be updated without interrupting its services.
6) State recoverability: Software deployment should have a mechanism to backup and restore the previous states of the software being updated.

C. Over the air software delivery

OTA delivery is a transmission of software from a provider to a mobile device over wireless networks such as Bluetooth or General Packet Radio Service (GPRS) [5]. In this work, the underlying OTA technique is employed as a means for deploying software.

D. Features of software deployment on mobile devices

1) Heterogeneity: Each device may operate under a different platform. As a result, the same software could not be deployed similarly on every mobile device. Software execution must satisfy the capability of the software with the device. For this reason, deployment of software on mobile devices must be adaptable to device capability.

2) Mobility and wireless connections: Because of mobility of mobile devices, users' context always changes. The demand for software may change according to the change of the context. Meanwhile, the users may not be aware of the availability of the software for their new context. Therefore, software deployment on mobile devices is required to support context awareness and wireless connections. While OTA user-initiated software deployment could not meet this requirement, a context-aware software deployment initiated by providers is a potential and suitable solution.

III. Related work

This section investigates previous approaches of software deployment on mobile devices. The approaches are analysed to identify their advantages and disadvantages.

A. Just-in-time software deployment

Just-in-time software deployment is one in which installation and activation is performed at the possibly latest time when users access the software. The examples of this approach are Java Web Star Technology [6] and Smart Deployment Infrastructure [7]. This method of software deployment saves memory of device, easily adapts to user preferences, device capability, and users' context. Moreover, updating software on mobile devices may be unnecessary because the software is not stored locally on the devices [3]. However, if the software is requested frequently, time consumption and network traffic would be high.

B. Component-based software deployment

A component-based architecture of software deployment on mobile devices was developed by [5], [7]–[9]. This architecture uses a functionality adaptation technique based on current context. The technique involves changing the way software carries out its functionality. This approach supports adaptation to user's context and device capability.

C. Middleware-based software deployment

Almost all approaches of software deployment have employed middleware architecture [10]. The architecture often consists of two modules. A module at the server is to handle users' requests, find correct software, and deliver software to devices. Another module at the client, the middleware is to download, install, and update software. Our framework uses this two-middleware architecture.

D. Context-aware software deployment

Context-aware software deployment employs context to determine functionalities or components being deployed to mobile devices [5], [8], [11], [12]. Our approach of software deployment adapts to users' interests by using users' subscription and adapts to device capability by matching device specifications with software specifications. The matching is to identify the compatible devices that will receive the corresponding software.

E. Pull model of software deployment

Pull model of software deployment allows users to discover software by themselves [10], [13]. OTA user-initiated provisioning [14] is an example of this model. In this model, an agent called Application Manager System (AMS) must be installed on mobile devices. AMS is used to discover, install, update, execute, and remove software on mobile devices. The pull model is suitable for software deployment on user demand. However, the users must check compatibility of the software with their devices. Moreover, the users may not be aware of the availability of new software as soon as the software is released. Similarly, when an update is released, there is a problem of how to notify the users of the new update.

F. Push model of software deployment

Push model or provider-initiation of software deployment allows providers to initiate a deployment process if they want [10], [13]. That is, new software or software updates can be pushed to mobile devices without users' discovery of the software or the updates. The model has been popularly applied for software deployment on desktop computers. However, its application for mobile devices has limitations. This paper adopts this push model for software deployment on mobile devices.

G. Summary and identification of problems

We has reviewed previous approaches of software deployment on mobile devices. We identified some problems remaining:

1) Because of heterogeneity of mobile devices, software deployment must adapt to device capability. However, the previous solutions have not significantly concerned this issue.
2) Providers-initiated software deployment has more advantages than user-initiated deployment. However, there is not a combination of these two models within the existing approaches.

Our solution for these problems is a new framework for software deployment on mobile devices that is a combination of push and pull models. Moreover, our framework is aware of device capability and users' interest. The design of the framework will be presented in the next section.

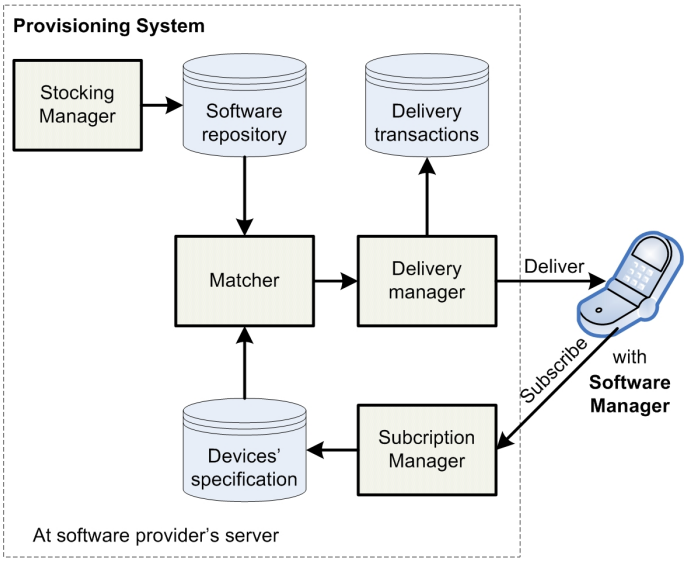

Fig. 1. Architecture of the OTA-PSD.

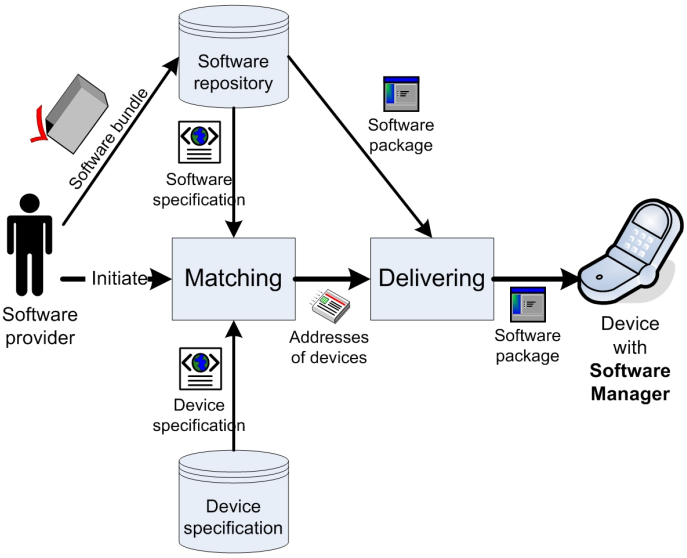

Fig. 2. Operation of the OTA-PSD.

IV. Design of the OTA-PSD

A. Overview

The OTA-PSD has two main modules as shown in Fig. 1. The *Provisioning System* is installed on the provider server. It handles requests for software, manages registration of software, notifies users of new software or new updates, and delivers software to mobile devices. This module has four main components including the *Stocking Manager*, the *Subscription Manager*, the *Matcher*, and the *Delivery Manager*. Another module is the *Software Manager*. It is pre-installed on the mobile device to provide the user with tools for subscribing to providers, receiving notifications from providers, download, installing software, and updating software. The *Software Manager* always listens to the providers for receiving notifications.

B. Design of the Provisioning System

The *Provisioning System* has four components.

1) Stocking Manager: This component manages registration of software with the software repository. Each software bundle consists of the software itself and the *software specification*. This specification includes the description of the software and the requirement of its execution environment (see an example in Appendix A).

2) Matcher Component: This component is responsible for matching device specification with software specification to identify the compatible mobile devices that will receive the particular software. This component is triggered to execute automatically once new software or software update is added into the repository.

3) Subscription Manager: This component manages users' subscription for receiving software from the provider. The *device specification* (see an example in Appendix B) is provided to the provider in the subscription process. The device specification is employed by the *Matcher* component.

4) Delivery Manager: This component manages transactions of software delivery. The *Matcher* component invokes this component to execute automatically. Before delivering new software to a mobile device, the *Delivery Manager* sends notifications of the availability of this software to the devices that the *Matcher* component has previously identified.

C. Design of the Software Manager

The *Software Manager* implicitly operates as an agent on a mobile device. It allows the user to subscribe to the *Provisioning System* for receiving software updates or new software of his/her interest. It continuously listens to the *Provisioning System* for receiving notifications from the provider. In normal conditions, the *Software Manager* is deactivated. Whenever a notification arrives at the device, the operating system on the device will automatically invoke it to execute.

D. Operation of the OTA-PSD

We present the operation of the OTA-PSD by describing the deployment of new software and a software update on mobile devices using this framework. Fig. 2 illustrates the operation of the OTA-PSD. Firstly, the software is registered to the software repository. This step uploads the software and its specification to the repository. Next, the *Matcher* component automatically matches the software specification with the device specifications to identify compatible devices with this software. Then, a notification about the software is send to the devices previously identified. Finally, if the user accepts to receive the software, the software will be downloaded and installed on the device.

1) Deployment of new software: As illustrated in Fig. 3, the process of software deployment has the following steps:

(0) Firstly, the user must subscribe to the *Provisioning System* if he/she has not previously subscribed. The *Provisioning System* will gather the user profile and the device specification by this procedure.

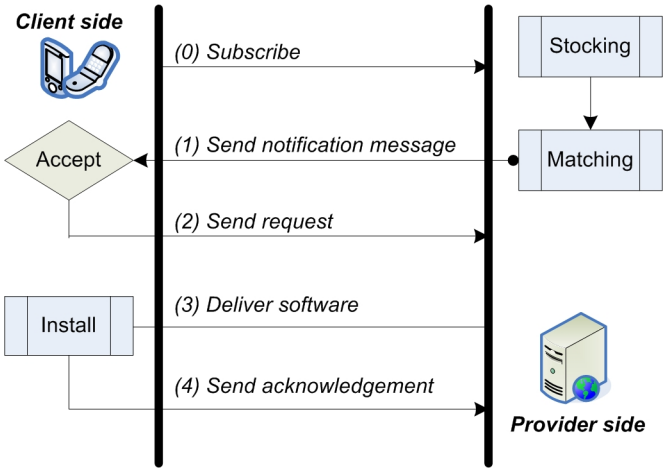

Fig. 3. Process of deploying software on mobile devices.

(1) When software or a software update becomes available on the server, the *Matcher* component will identify compatible devices with the software. Next, the *Delivery Manager* will sends notifications to these devices.
(2) When the notification arrives at the device, the *Software Manager* on the device is automatically invoked to inform the user. If the user accepts to download this software, a request is sent to the *Provisioning System*.
(3) Next, the software is delivered to the device.
(4) After receiving the software, the *Software Manager* installs and configures the software. Optionally, an acknowledgement is sent to the server to confirm the success or the failure of the installation.

2) Deployment of software updates: There are two phases for updating software based on the OTA-PSD. In the first phase, the software update and the software specification are uploaded to the repository. The older version of the software will be maintained in the repository. Next, the *Matcher* component will investigate the specification of the update. It uses device specifications and previous delivery transactions in databases to identify suitable mobile devices. These mobile devices will be notified of the new update. The *Software Manager* on the devices will handle remaining tasks such as downloading and installing the update as presented above in the process of deploying new software.

V. Implementation of the OTA-PSD

We implemented the OTA-PSD system called the deployment system. In this section, we describe the implementation of this system in detail.

We choose Java as a programming language to implement the system. The system has two modules. The first module is called the *Provisioning System*, which is a web application running on Apache Tomcat web server [15]. The second module is called the *Software Manager*, which is installed on mobile devices. We use the object-oriented architecture for modelling the system. This architecture makes the components of the deployment system reusable, maintainable, and testable.

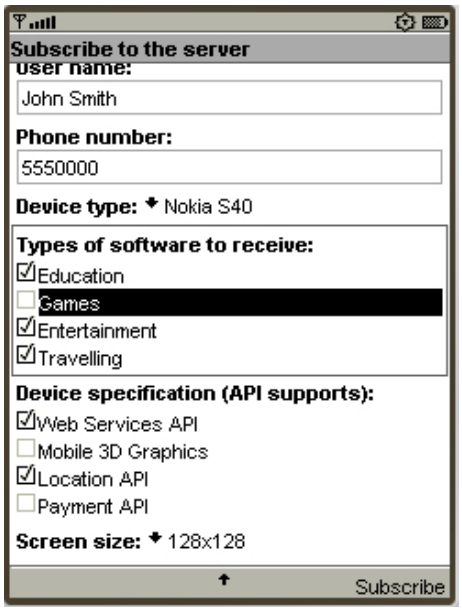

Fig. 4. The *Software Manager* is used to subscribe to the provider in order to receive new software of the user interest.

VI. A case study

We developed an example application called Holiday Planner, which helps users to plan their holiday. This application is to evaluate the feasibility of the OTA-PSD and simulate the operation of the deployment system based on the OTA-PSD. Firstly, we present the deployment of the example application as a simulation of deploying a new application on a mobile device. Secondly, we assume that the example application is upgraded to a new version. So, we describe the deployment of the new version on the mobile device. We employed the use of Sun Java Wireless Toolkit 2.5.1 in our simulation.

A. Deployment of the example application

1) User subscription: As showed in Fig. 4, we utilize the *Software Manager* on our mobile device to subscribe to the *Provisioning System*. We use the mobile phone number as an address for receiving notifications afterwards. Naturally, our system allows to use other kinds of addresses such as IP address and Bluetooth address.

2) Software registration: The example application is registered to the repository by using the *Stocking Manager*. After that, our device is notified of the availability of the application.

3) Processing on mobile devices: When our device receives the notification, the *Software Manager* on the device is automatically invoked. The *Software Manager* displays the information about the application on the screen as in Fig. 5. If we accept to download this application, the application will be downloaded and installed automatically.

B. Deployment of the new version of the example application

Because the OTA-PSD is designed to support monolithic software, the deployment of updates is similar to the deployment of new software. Indeed, updating software in our

Fig. 5. The information of the new software is notified to the user. It asks the user a confirmation of downloading and installing the software.

framework is the replacement of the previous version of the software by the new one. For the purpose of demonstration, firstly, we improve our example application to the new version. Next, the new version is re-registered to the repository. Then, the *Provisioning System* identifies the mobile devices, which has previously installed this application. After that, the system notifies the corresponding users of this new version. Finally, if a user accept to update the example application on his/her device, the new version will be downloaded to replace the previous version.

VII. Evaluation

The successful demonstration as presented in the previous section shows the feasibility of the OTA-PSD. The OTA-PSD has some advantages, which are different from existing approaches. In our framework, the deployment of software can be initiated by providers. Thus, software can be deployed based on users' context such as users' location. Moreover, the deployment process in the OTA-PSD takes device specification into account. Therefore, the deployment process can adapt to device capability. In this section, we discuss and evaluate the advantages and disadvantages of our framework.

A. Advantages of the OTA-PSD

1) Provider-initiated deployment: One of the design objectives of the OTA-PSD is provider-initiated deployment. By this model, users do not have to be concern about discovering new software or software updates. To receive software of the interest, users only need to subscribe to the provider. Moreover, updating software is carried out automatically with the minimum of users' intervention. Therefore, users can save on time consumption for discovering and updating software. Consequently, this approach enables providers a possibility of deploying software or services based on user's context.

2) Adaptation to device capability: Adaptation to device capability is a special requirement for software deployment on mobile devices because of the heterogeneity of mobile devices. The heterogeneity includes the variety of hardware capabilities, the variety of screen sizes and API libraries that are supported by mobile devices. To enable our framework to be aware of device specification, we implemented a component called the *Matcher*. Before deploying particular software, the *Matcher* is invoked to identify the compatible mobile devices, which satisfy the software requirements. As a result, the compatible devices will receive this software.

B. Disadvantages of the OTA-PSD

In our solution, a user must pre-define the specification of the device. Therefore, once the device are upgraded or modified, the user must re-declare the modification of the device to the deployment system. Moreover, because the OTA-PSD only supports monolithic software, updated software will replace entirely the pre-installed software with the new version. Therefore, in some situations, although the software needs to be updated in small segment, the entire new version of the software must be completely downloaded and installed. As a result, the update process may consume more bandwidth than the approaches of component-enabled updates.

Although the software deployment is initiated by providers, allowing the automatic update on mobile devices may bring some risks to the users because our framework has no rollback mechanism when conflicts or issues arise. Another restriction of our approach is the use of mobile phone numbers, Bluetooth addresses, or IP addresses of mobile devices to determine a particular device. As a result, if the addresses are changed or used by another device rather than the original device, which has previously subscribed to the provider, the deployment system will not recognise these changes unless the user re-subscribes to the provider. If the user does not re-subscribe, he/she may receive the software which is not compatible with his/her device or not of his/her interest.

VIII. Conclusion and Future work

This work has investigated previous approaches in the field of OTA software development on mobile devices. Consequently, we have proposed a new framework for OTA provider-initiated software deployment on mobile devices called OTA-PSD. Our approach to software deployment on mobile devices is a combination of push and pull models of software deployment. Moreover, our framework can be aware of device capability and users' context. We have developed a deployment system based on the OTA-PSD. Then, we have proved that this system operates properly and meets our design objectives of the OTA-PSD. Finally, we have implemented a case study to simulate the operation of our deployment system. In the case study, we have used our deployment system to push an example application, which is called Holiday Planner application, to a mobile device without the user discovery of this application.

Despite achieving satisfactory results, the research has some limitations. It has not a mechanism for evaluation of bandwidth, time, and memory consumption for a session of software deployment. In additional, our framework only supports deployment of monolithic software. Therefore, updating software based on the OTA-PSD has been remaining some restrictions in terms of bandwidth consumption.

In future work, we will improve our framework to automatically identify device specification, to support component-based software deployment. We will also develop a mechanism for rollback when conflicts or issues occur during updating process. In another direction, we will apply the OTA-PSD to deploying mobile web services on mobile devices [16].

APPENDIX A
AN EXAMPLE OF SOFTWARE SPECIFICATION

```
<software name="Holiday Planner"
   version="1.0">
   <description>
     Searching for cheapest flight
   </description>
   <vendor>ABC Company</vendor>
   <size>184KB<size>
   <price>50USD</price>
  <type>education</type>
   <requirements>
       <require>
           <name>screensize</name>
           <value>any</value>
       </require>
       <require>
           <name>colordepth</name>
           <value>1</value>
       </require>
       <require>
           <name>bandwidth</name>
           <value>3M</value>
       </require>
       <require>
           <name>platform</name>
           <value>Java</value>
       </require>
       <require>
           <name>CLDC</name>
           <value>1.0</value>
       </require>
       <require>
           <name>MIDP</name>
           <value>2.0</value>
       </require>
       <require>
           <name>webservicesAPI</name>
           <value>true</value>
       </require>
   </requirements>
</software>
```

APPENDIX B
AN EXAMPLE OF MOBILE DEVICE SPECIFICATION

```
<device typename="Nokia 6060">
   <supports>
       <support>
           <name>screensize</name>
           <value>240x320</value>
       </support>
       <support>
           <name>colordepth</name>
           <value>3</value>
       </support>
       <support>
           <name>platform</name>
           <value>Java</value>
       </support>
       <support>
           <name>CLDC</name>
           <value>1.0</value>
       </support>
       <support>
           <name>MIDP</name>
           <value>2.0</value>
       </support>
       <support>
           <name>webservicesAPI</name>
           <value>true</value>
       </support>
       <support>
           <name>locationAPI</name>
           <value>true</value>
       </support>
   </supports>
</device>
```

REFERENCES

[1] S. Jamadagni and M. Umesh, "A PUSH download architecture for software defined radios," in *IEEE International Conference on Personal Wireless Communications, 2000*, Hyderabad, Dec. 2000, pp. 404–407.

[2] A. Heydarnoori, F. Mavaddat, and F. Arbab, "Towards an automated deployment planner for composition of web services as software components," *Electronic Notes in Theoretical Computer Science*, vol. 160, pp. 239–253, Aug. 2006.

[3] D. Ayed, C. Taconet, G. Bernard, and Y. Berbers, "An adaptation methodology for the deployment of mobile component-based applications," in *ACS/IEEE International Conference on Pervasive Services, 2006*, Jun. 2006, pp. 193–202.

[4] S. Ajmani, B. Liskov, and L. Shrira, "Modular software upgrades for distributed systems," in *European Conference on Object-Oriented Programming (ECOOP)*, Jul. 2006.

[5] T. Fjellheim, "Over-the-air deployment of applications in multi-platform environments," in *Proc. ASWEC'06*, 2006, pp. 159–170.

[6] Sun Microsystems, Inc. (2007, April) Java web start technology. [Online]. Available: http://java.sun.com/products/javawebstart/

[7] C. Taconet, E. Putrycz, and G. Bernard, "Context aware deployment for mobile users," in *Proc. COMPSAC '03*, 2003, pp. 74–81.

[8] N. M. Belaramani, C.-L. Wang, and F. C. M. Lau, "Dynamic component composition for functionality adaptation in pervasive environments," in *Proc. FTDCS '03*, San Juan, Puerto Rico, USA, May 2003, pp. 226–232.

[9] M. Kwan, "A distributed proxy system for functionality adaptation in pervasive computing environments," Master's thesis, Department of Computer Science and Information Systems, The University of Hong Kong, Aug. 2002.

[10] L. Sobr and P. Tuma, "SOFAnet: Middleware for software distribution over internet," in *Proc. SAINT '05*, 2005, pp. 48–53.

[11] S. Ali, T. Torabi, and H. Ali, "A case for business process deployment for location aware applications," *International Journal of Computer Science and Network Security*, vol. 6, no. 8, pp. 118–127, Aug. 2006.

[12] Y. Weinsberg and I. Ben-Shaul, "A programming model and system support for disconnected-aware applications on resource-constrained devices," in *Proc. ICSE '02*. ACM Press, 2002, pp. 374–384.

[13] M. Mehta, N. Drew, G. Vardoulias, N. Greco, and C. Niedermeier, "Reconfigurable terminals: an overview of architectural solutions," *IEEE Commun. Mag.*, vol. 39, no. 8, pp. 82–89, Aug. 2001.

[14] C. E. Ortiz. (2002, Nov.) Introduction to OTA application provisioning. [Online]. Available: http://developers.sun.com/techtopics/mobility/midp/articles/ota/

[15] The Apache Software Foundation. (2007, Apr.) Apache tomcat. [Online]. Available: http://jakarta.apache.org/tomcat/

[16] P. Farley and M. Capp, "Mobile web services," *BT Technology Journal*, vol. 23, no. 3, pp. 202–213, 2005.