Senior Design
Project 2008/2009

Mobile Project Tracking Program

Jon Deokule
djon@seas.upenn.edu

Advisor: Zachary Ives
Table of Contents

Abstract........................................................................................................................................... 3
Related Work ................................................................................................................................... 3
Technical Approach............................................................................................................................ 4
  System Overview ............................................................................................................................ 4
  Web Services in the Domino Database.......................................................................................... 5
  Communication – Mobile Data System ......................................................................................... 5
  Device Application ......................................................................................................................... 7
Development Environments ............................................................................................................. 7
  Development .................................................................................................................................. 8
  Web Services .................................................................................................................................. 8
  Device Application ......................................................................................................................... 8
User Features.................................................................................................................................... 9
Conclusion........................................................................................................................................ 10
  Accomplished and Unaccomplished Work .................................................................................. 10
  Challenges and Takeaways ........................................................................................................... 10
Bibliography ..................................................................................................................................... 12
Abstract

It is unarguable that technology, specifically software, has become an essential contributor to the bottom line of firms. However most products target large corporations and have few feasible and affordable solutions for smaller businesses. Two years ago I participated in the design and development of a job management program to manage the construction of new homes – scaled and tailored to a small firm. It is built on a Lotus Domino database and has a front end user interface in Lotus Notes. However, this is a construction company where employees are frequently in the field – with no access to a desktop client.

This project sought to mobilize the capabilities of this existing job management system by building a native application on the BlackBerry OS. The mobile client connects to the existing system and mirrors the functionality of its desktop counterpart. Issues specific to mobile development such as network issues, security, and integration in the existing system needed to be considered.

Related Work

The core goal of this program is to manage progress and status of all jobs of a small residential construction firm. There are some solutions which accomplish part of this goal for some subset of the target users.

There are currently several firms which develop software to manage residential construction jobs and projects. These solutions target firms with >5,000 employees with thousands of houses or buildings completed annually. They provide a rich feature set for managing all of these jobs and offer immense scalability for future expansion. These extensive capabilities are too powerful and expensive for the target users of small firms.

Streamlined versions of these large products also exist in various forms. Some firms use laptops with the same program in order to be mobile. This model is inefficient because laptops do not make the users as mobile as possible. It also provides field users too many features that aren’t needed when mobile. There are companion applications for the Blackberry OS or Windows Mobile OS which complement the existing products. However the flaw of these products is similar to the desktop-based version – they provide too many features and are too expensive for the smaller firms.
This proposed project will extend the features of the job tracking program that is already developed. It will be optimized in two ways – functionally and technically. Functionally the user interface will be optimized to display only the information the user needs when mobile in the field. Technically the fundamental principles will be similar to the existing products. All of these programs which have mobile components account (at some level) for offline use and other data issues encountered with over the air transmissions.

It is necessary to have the application on the BlackBerry OS since those devices are currently used by this company. BlackBerry currently does not have anything solution which specifically can accomplish this task (BlackBerry, 2009).

Technical Approach

System Overview

The current system comprises of a Lotus Domino database for the back end and an application running within Lotus Notes on the front end. Lotus Domino and Notes are similar to Microsoft Exchange and Outlook. The database contains the tables, views, information, and logic used in the system. It also contains the desktop application which runs on the front end.

Since the new BlackBerry system uses a client server model the technical approach had to be considered from back end to front end. The main system components are the back end Domino database, two-way communication, and the device application. Each of these components needed to be built and integrated.

The data contained in the database was previously only available through the Lotus Notes interface. An entirely new method for other systems and applications to request and receive database information needed to be constructed. This was accomplished through web services. After the information was out of the database it needed to reach the handheld. Reliability, security, and ease of integration and implementation needed to be considered when evaluating multiple options. BlackBerry’s Mobile Data System (MDS) was chosen for communication because it satisfied all the conditions. BlackBerry devices also have multiple runtime environments available. The MDS runtime environment was chosen for its easy integration with MDS communication.
**Web Services in the Domino Database**

Web services are built to make the data in the application available to external applications. A web service is an archive of remote operations which can be used by calling them over the Internet (IBM, 2004). A WSDL (web service description language) outlines the available methods, their parameters and types, and the returned data and type. This file is XML formatted and published on the web for consumers. Calls and corresponding data returns are done over HTTP using the SOAP messaging format. SOAP messages are formatted in XML and are a standard protocol for formatting messages used over HTTP (IBM, 2004).

When dealing with corporate data security is a concern. While the WSDL file and methods are published online, they are by no means insecure. The Domino database maintains a list of authorized users and groups to use the application. The Notes client uses this information to grant users access on the desktop. When connecting to a web service the same authentication process is used, prompting the user for credentials before granting any access.

Behind the web service is code to perform operations on the database and Domino supports Java and LotusScript. Java was predominantly used in this system. Lotus provides a lotus.domino jar file which is its API for the database. Java objects represent all aspects of the database from the database itself, database sessions, and documents contained within. By using the API in the Java classes the underlying query calls are abstracted. The API also provides error checking and exception handling when accessing the database, making it manageable to write the web service in Java code. The classes are compiled and published all from within the database. The web service abstracts the classes themselves so the code and other private information remain secure and unseen to the consumer.

The development of the web service on this system has large potential extending past the BlackBerry client in this project. It allows for a myriad of future client developments using this service and database. In the future client applications on other platforms such as the iPhone, Windows, Windows Mobile, web, Mac, and Linux could all be developed using this same service.

**Communication – Mobile Data System**

The web service back end is accessible through HTTP, which the device would be able to connect to directly. However this model is unsecure, especially when dealing
with corporate data and firewalls. Additionally multiple applications on the phone all making simultaneous connections can degrade performance.

The BlackBerry Mobile Data System is an entire communication framework provided by BlackBerry. It provides a secure means for data transmission between the device and back end. A corporate BlackBerry device maintains a secure connection to the BlackBerry Enterprise Server (BES), sitting behind a company firewall. The BES pushes mail and other data to the device and handles most network connection issues. Figure 1 outlines the communication model (BlackBerry, 2009).

![Figure 1](image)

The BlackBerry MDS server runs within the BES. It communicates to the devices through the secure connection established by the BES. If multiple applications on a device use MDS communication, even to different web services or application server, all the data is transmitted through this one connection. This many to one design generates far less overhead than by multiple connections and ensures secure transmission.

When the application makes a web service call the parameter data is put in a BlackBerry message and sent to the MDS server. The MDS server will decrypt and unpack the message. Then the MDS server (not the device) will invoke the web service call. Since the MDS server invoked the call, it receives the response. This response is then packaged and encrypted in a BlackBerry message and pushed to the device. If the web service or application server is on the same network as the BES (as shown in figure 1) the data doesn’t leave the corporate intranet, thus adding another layer of security. The HTTP protocols and message format (XML, SOAP, etc.) used by the application server are abstracted from the handheld.
When transmitting data over the air there is a chance that a connection will be interrupted and the data will not be fully and accurately transmitted. Blackberry’s MDS system ensures data transmissions in both directions. If a device is off or does not have service and a message is sent to the device, it gets queued until the device is available. This established system will handle data verification issues and abstract the network protocols.

**Device Application**

The BlackBerry OS contains a MDS runtime environment which provides common services and a framework on which all MDS applications can run. This framework handles receiving MDS messages and ensuring the correct application receives them. MDS applications are not built on classes but in various screens and scripts. Scripts are written in JavaScript and can be used for tasks such as data processing and screen flow.

The application flow and actions are implemented in several different ways. Controls (such as buttons) and events (such as receiving a message) trigger actions or scripts. A button control can be assigned to open a different screen or run a script. A combination of scripts and actions were used to complete this application. After receiving data from the server scripts are used to process the data and put it into data types defined on the phone.

**Development Environments**

Several steps were taken to provide an isolated and safe environment to be used for development and testing. Any number of problems could arise due to bugs and there cannot be downtime in the active servers and database. A copy of the Domino database was made and a separate instance set up dedicated for development. The web services were programmed and tested in this instance before being deployed to the live server.

Unlike the Domino server which can create several instances, the BlackBerry Enterprise Server cannot. A virtualized Windows Server 2003 was setup and a BES with MDS server was deployed. This isolates the live BlackBerry server and ensures no downtime for the current users. The application development only required a program so no additional steps were needed.

Since one the operating systems is running on virtualized software that uses NAT networking. This makes it difficult to establish connections with IP addresses over the
Internet. LogMeIn’s Himachi product was used in order to facilitate secure communication between all three. This free software establishes a VPN between the computers. Each is assigned an IP in the VPN which remains static. Communication between Domino server, BlackBerry server, and the BlackBerry development environment is all done over this VPN.

Development

Web Services
The web services are written in Java and LotusScript and reside on the Domino server. Domino Designer is an application provided by IBM to design, configure, and program many aspects of a Domino database, including web services. The web service classes were created and compiled from within Domino Designer. Changes are saved locally and replicated on the server. Back end debugging output is also viewed from within Domino Designer.

Device Application
Blackberry provides an IDE for MDS application development – MDS Studio. It is a comprehensive IDE which contains several features to construct the entire application. GUI screens, request messages, response messages, data mapping, and object definitions are done through MDS Studio.
Figure 2 shows a snapshot of the GUI environment to construct the BlackBerry’s GUI screens. Common GUI objects such as labels, text boxes, radio buttons, etc. are available with a drag and drop design. The objects’ displays can be populated when the screen loads, on receipt of a message, or refreshed with scripts.

BlackBerry supports primitive data types however more complex objects are often needed. Data type definition is also done within MDS Studio. For example, in this project a “job” data type was needed to hold fields related to a job. Objects can be global variables, local variables to screens, local variables in scripts, or parameters passed between screens. Request and response messages are also created here. Message data fields are defined which correspond to parameters and returns in the web service. Data mapping is implemented to have the message field use data from a function, global variable, or user input field.

User Features

In addition to the functionality already in the system, this project was going to add two new user features specific to the mobile device. The first was the ability to leverage GPS information from the phone. By knowing its position it could automatically know which job it was at and open the appropriate record or sort all jobs by distance. The second feature was the ability to take a picture on the phone and upload it to the job record on the server.

Due to unforeseen circumstances neither feature was able to be implemented in the final version. In order for GPS lookups of jobs to work there needs to be GPS information in the job’s record. The system currently does not maintain this information so all the GPS information for all the jobs would need to be obtained. This would require back end modifications and GPS lookups for all jobs which was outside the scope of this project. Additionally the GPS positioning is not always accurate and can be off by several blocks. Since many jobs are clustered together and adjacent to each other this would yield inaccurate results when automatically opening job records.

There are currently no pictures stored in job records. In order for the picture upload feature to work database modifications would be needed on the back end. A third-party developer maintains the database so these modifications would be costly and outside the scope of the project (Greenberg, 2009).

A large aspect of user features and usability is the presentation of the information on the screen in the most efficient way. Alpha and beta testing with end
users on the device is essential in creating a UI that is useful to the user. Development in Philadelphia and end users (as well as all active BlackBerrys) in New York significantly prohibited this aspect of testing and refinement. A basic UI is constructed for the initial deployment. Since all framework is in place for getting data from the server to the device, changes to GUI appearance are minor and easy to implement in the future.

**Conclusion**

**Accomplished and Unaccomplished Work**

This project has changed slightly since the original proposal however the majority of work as outlined in the proposal was accomplished. As more research was done at each step, the tasks and challenges became clearer. The overall goal of mobilizing the existing desktop application on a BlackBerry was completed.

The back end server code can successfully interact with the database. There are no known bugs or issues because of extensive testing. Any errors on the back end are handled and do not cause a crash on the server nor the handheld. The web service is currently completed and deployed on the live Domino server. The MDS server on the BES was not used prior to this project. It has been configured and setup and is currently live on the BES.

The client application is completed and basic functionality is working on the handheld. The two main mobile features (GPS location and picture uploading) were not completed. Circumstances outside the scope of this project prevented the development at this point in time but are still possible in the future. All the back end work of bringing data to the device is done but the layout on the screen was difficult complete without the end users present. Currently only simple views of the basic information are done. Editing GUI screens and placing information on the screen is simple so updates would require minor work.

**Challenges and Takeaways**

Nearly everything related to the development of this project was new to me. No previous projects or experiences required back end server development or communication between a client and server. Virtualizing and configuring a new server installation, installing and configuring BES, and setting up a web service were all new and required research and experimentation from the ground up. Designing and planning was incredibly difficult because there was no basis to gauge the difficulty of
time of a given task. The great majority of time spent on this project was learning and navigating the tools and frameworks available to accomplish a task.

There were several benefits from this project. The deliverable is a real, working application and system that will be used in production. Personally, I finished this project with an entirely new skill set. First I have a good idea of the general components of a client-server system which can be applied when examining or building other systems. Additionally I am fairly proficient in developing an application for the BlackBerry OS. 90% of the time spent on this project was spent learning the systems and protocols. With this overhead completed, the next system can be done with increased sophistication and ease.

This project was more than an academic exercise; it was designed to be a real world, usable application that will provide added productivity to a business. Users will have version 1 deployed on their BlackBerry handhelds by the end of May 2009. Feedback on the GUI and user experience will start immediately after deployment. Additional screens, refinement of current views, and updates to the GUI will continue after the completion of the project.
Bibliography

