SOFTWARE DEVELOPMENT ISSUES OF A DISTRIBUTED MOBILE-COMMERCE SYSTEM

Phuong D. Nguyen, Christian Kern, Christian Wattinger, Martin Guggisberg, Helmar Burkhart, Peter Maier*
Computer Science Department, University of Basel
50 Klingelbergstr, CH-4056
* All Internet GmbH
Switzerland
Phuong.Nguyen@unibas.ch

ABSTRACT
The proliferation of mobile devices such as smart phones and Personal Digital Assistant (PDA) opens new ways for developing mobile E-commerce systems (so called Mobile-commerce or M-commerce system). In M-commerce systems, we still see many common components that we have in standard E-commerce applications, such as web servers and database servers. However, these new applications raise some unique challenges. For example, the limitations of mobile devices (e.g., small screen size and reduced CPU performance) implies that software development is partly different from desktop applications. Questions which are posed when designing M-commerce system include: “What kind of information do the users want to access via mobile devices?”, “How can we provide useful applications with limited user input facility?” “How can we test a distributed M-commerce system?”

In this paper, we discuss about some software development issues for mobile commerce systems from our experiences while developing an M-commerce system for tourists. Our prototype, easyHotel, is an useful software that allows booking hotel rooms via mobile phones.

KEY WORDS
Distributed E-commerce, software development, object-oriented systems, software reuse.

1. Introduction

Mobile devices are very popular. According to a report of the German Ministry of Economics and Labour [1], in year 2004, the number of mobile phones is nearly two times the number of PCs in the world (1495 Millions mobile phones vs. 822 Millions PCs). Also in this report, every second person in Europe has a mobile phone, and in US the percentage is 34 mobile phones per every 100 persons. A basic service of mobile phones systems, the short message service (SMS) is growing significantly and rapidly [2]. There is an increasing number of mobile commerce applications in various areas such as Banking [3-4], Restaurant (make order and payment) [5], Mobile Payment (e.g.: buy food from vending machine, or tram ticket) [6]. However, software for mobile devices is not just a down-scaled desktop version. When designing M-commerce systems, some issues should be considered:

• What kind of information do the users want to access via mobile devices?
• Physical limitations of mobile devices (e.g.: limited memory, CPU performance, and user input facilities): What kind of software architecture is suitable?
• How can we add a “mobile component” to an existing E-commerce system?
• How can we test a distributed M-commerce system?

Taking all these issues into consideration, we implemented a system for hotel reservation using mobile devices. Online room reservation is not a new idea. There are numerous of such web applications. However, a hotel booking system using mobile devices is quite new. To our knowledge, there are some systems to book a flight [7], or to book a taxi [8] via SMS. However, there only exist a few simple systems that send SMS alerts to travellers to confirm hotel bookings and to remind them several hours prior to their check in time [9]. The use of mobile device clients for hotel booking offers several advantages over web-based clients. Firstly, one can not assume that everyone would always use a web-based booking system because the users might not have access to a PC and an internet connection. For example, in case of late arrival or temporary plan, it is not easy for tourists to find a PC-internet service in a city that they don’t know before. Young people travel very much, but they often have temporary plans. They would visit a city if they find a cheap way to do so. Secondly, searching on the Internet can take a lot of time and does not always give the best result. It is not unusual that after hours searching the Web, users only found some expensive hotels.

Our system currently uses SMS for booking the rooms but other client types such as WAP and J2ME can easily be added. The main benefits of using our system are:

• Straight-forward functionality: a customer only needs to send 3 SMS in order to search, and book a room.
• The hotel and tourists can both get benefit: low price for the tourists (fixed last minute price) and maximum room usage for the hotel.
• Easy to use: requires no time to learn the system as many people have a mobile phone and know how to send an SMS.
• Independent from time and place: the users can book their rooms when they want, no matter where they are. Access to a PC and Internet is not required.
• The room database is up to date (updated every day).
• The use of widely accepted standard technologies and open source software: our system use only standard and open source components. The system can run on any hardware platform. The minimum configuration requires only a normal Web server and a mobile phone.

(See Section 3.2 for detailed list of software components).

The structure of this paper is as follow:
In section 2, we will discuss about the answers of the above mentioned questions. Specifically, we discuss in detail some important issues such as thin client approach, testing tool for distributed system. In section 3, we describe our system architecture. Finally, the last section presents conclusions.

2. Software development issues

In this section we discuss about software requirements of M-commerce systems in general. Of course, some issues depend on the size of the project (e.g.: world-wide active enterprise or regional small and medium sized company).

2.1 Interoperability and extensibility

A questions which is often asked by developers of M-commerce systems is “How to add a “mobile component” to an existing E-commerce system?” It is not surprising that XML is an important part of the answer. Mobile devices have limited bandwidth and processing power. Therefore, exchanging text-based data is more efficient than object-based data. In addition, the use of XML ensures that the system can easily interoperate with another existing E-commerce system.

2.2 Concurrency and scalability

The requirements of concurrent access (serving many users at the same time), and system scalability make Java a very good candidate. Besides the advantage of platform independence, Java provides concurrent access and scalability through its powerful class libraries, with moderate programming effort. It is also very easy to work with XML in combination with Java using Java built-in classes and libraries for Java-to-XML binding such as Castor [10].

The code snippets (a simplified version of our system implementation) listed below demonstrate how simple it is to use the Castor framework for binding of Java classes to XML documents and vice versa.

If we want to map the following class:

class Customer{
    private String ID, firstName, lastName;
    public Customer() {
        firstName = "";
        lastName = "";
        ID = String.valueOf(System.currentTimeMillis());
    }
}

to an XML document like:

<customer>
    <ID>0726336123</ID>
    <firstName>Andreas</firstName>
    <lastName>Schneider</lastName>
<customer>

we might use the following Castor mapping file:

<class name="CH.unibas.cs.eh.castor.Customer">
    <map-to xml="customer"/>
    <field name="ID" type="java.lang.String" get-method="getID" set-method="setID"/>
    <field name="firstName" type="java.lang.String" get-method="getFirstName" set-method="setFirstName"/>
</class>

The following class demonstrates how to use Castor to manipulate the file “customer.xml”. It unmarshals the document “customer.xml”, set the customer’s firstName in a java object and marshals the object model back into XML with the new first name.

```java
import org.exolab.castor. *; public class Demo {
    public static void main(String args[]) {
        Mapping mapping = new Mapping();
        try {
            // 1. Load the mapping information from the file
            mapping.loadMapping( "mapping.xml" );

            // 2. Unmarshal the XML data into a java object
            Unmarshaller unmarshal = new Unmarshaller(mapping);
            Customer aCustomer = (Customer)unmarshal.unmarshal(new InputSource("customer.xml"));

            // 3. Change the first name
            aCustomer.setFirstName("Alice");

            // 4. marshal the object to XML data with the new first name
            Marshaller marshaller = new Marshaller(new
                OutputStreamWriter(System.out));
            marshaller.setMapping(mapping);
            marshaller.marshal(aCustomer);
            //end try
            catch (Exception e) {
                System.out.println(e);
                return;
            } //end catch
        } //end main()
    } //end class Demo
```
2.3 Thin client approach

Mobile devices have limited bandwidth and CPU power. Therefore, it is advisable to move the data processing burden to the server side and let the thin mobile clients do simple tasks such as presenting data and gathering users' input.

In our M-commerce application (hereafter referred to as easyHotel), the Model-View-Controller (MVC) paradigm is used. MVC is a well-known design pattern that provides several benefits. MVC makes the application more easily modifiable by separating data persistence, data presentation, and control flow into different components. Moreover, collaboration is done through clearly defined interfaces. For example, in M-commerce applications, new client types are easy to add by adapting the new client to operate as an MVC View. In such case, the other components don't have to be change. Thus, software reuse is achieved as well as system stability (less bugs in code because we still use the tested components: Model and Controller when adding the new client types).

Our current prototype of easyHotel uses SMS as the View component. However, more powerful client types such as WAP and J2ME can easily be added. We decided to use SMS as the first client type because it is fast and easy to use SMS. Within easyHotel, only 3 SMS are needed in order to search and book a room. Furthermore, anyone who has a mobile phone knows how to send an SMS. Moreover, the current limitations of mobile devices (in memory and CPU power) require simple user interfaces. J2ME applications provide a more powerful GUI (with button, list, textbox and so on) but it takes time for users to learn the software. However, we are planning to support J2ME clients in the next easyHotel release as now many new mobile phones support J2ME.

2.4 Testing distributed M-commerce systems

There are many different topics related to software testing, for examples: black box test, white box test, unit test, automated test, etc. Discussions about these topics are out of the scope of this paper. In this section, some experiences learnt while developing our prototype will be discussed:

It is mandatory to use an automated testing tool

There are a number of automated software testing such as JMeter [11] and IBM Functional Tester [12]. They can be used to test a software system in many different ways. For example, in a Web application, software developers want to see if application performance is still good when a lot of requests come at the same time (say 10,000 requests). It is very difficult if not impossible to test such a scenario manually. With a testing tool, we can easily write a testing script, specify how the parameters are changed and run the script. In this way, thousands of requests are sent concurrently and we can repeat the test as often as we want. Finally, several server performance metrics (throughput, average of request/response time, etc) are displayed graphically.

There are several advantages of using automated testing software such as:

- Simulate a large number of users.
- Test the system on different input: the request's parameters can be changed by program.
- Repeat testing many times.

Testing with real devices is very important

We used JMeter, an open source testing software. During the development phase, we tested our server application intensively with hundreds of concurrent requests. The result was quite promising: on a moderate hardware configuration (CPU: 3 GHz, 1GB RAM; Nokia phone 7110), the system can serve around 900 requests per second and the average request-response time is only a few seconds. Interestingly, when we tested with real devices (specifically, we used 3 mobile phones to send concurrent SMS), the response time was very long (few minutes). It was quite complicated to find out the reason because no error had occurred. The system just responded slowly. Did the testing tool give us a wrong result? Fortunately, the answer is No. When we carefully considered the testing scenario, we found out the reason: The system works as follow: the user sends a request (SMS) to the SMS gateway. The gateway parses the SMS and makes an HTTP request to the server. The server processes the request and sends back the result to the SMS gateway. The gateway sends the SMS to the user. Technically, SMS are sent from the mobile phone to the base station through radio signals. It is difficult to simulate this process. Therefore, we only simulated part of the process (started from the second step: the gateway sends requests to the server and so on). The reason was that the SMS gateway queues incoming SMS. When many SMS come at the same time, only the first SMS is processed, the other SMS are saved in a queue and they will be processed at a later time. Of course, we can adjust the waiting time in the queue. However, if we did not set this parameter, it is set to several minutes by default. That's why when we sent 3 SMS concurrently, the response time was several minutes. After we reduced the queuing time to a few seconds, the performance was improved noticeably and was comparable to the test results.

Figure 1a shows the main user interface of JMeter. On the left panel, there are several options to see the server performance. We can show the result in several forms such as: Diagram for request's time with maximum and minimum value (Spline Visualizer); Table that list all the results. Throughput (the number of requests/minute the server handled); Average of request-response time of all samples as well as the standard deviation and the median value (in milliseconds).
Figure 1a: User interface of JMeter testing software.

Figure 1b: View server performance results graphically in JMeter.
3. easyHotel Application

3.1. Overview
easyHotel allows a tourist to search for a hotel and to book a room using a mobile phone. The system works by sending an SMS to a special mobile number with a pre-defined keyword and/or some additional information (e.g.: city name, zip code, etc). In addition, easyHotel can also be accessed through web interface as a standard online booking system. There are web interfaces for:
- Customer (tourist): to book a room and manage the bookings.
- Hotel staff: each hotel will have an account by which hotel staff can insert, change and delete the available rooms. Hotels can see detailed information about how many rooms are booked, a statistical report of bookings in a day, week and month.
- Administrator: the administrator manages the easyHotel system: add a new user, add a new hotel account, database maintenance, etc.

3.2 Technologies used
Platform independent, widely accepted and standard technologies, open source software are important criteria that we considered in the system design phase. Specifically, easyHotel uses the following technologies:
- Standard Web technologies (HTTP, HTML)
- Java (Servlet, Java Server Page) [13]
- Open source servlet container (Tomcat) [14]
- Open source SMS gateway (Kannel) [15]
- Open source XML database (Xindice) [16]

3.3 System architecture
Figure 2 presents the system architecture. The system is composed of 3 main components:
- The View (presents the data and collects the users input): as shown in the figure, the system supports several client types such SMS, WAP, normal web browsers (web contents dynamically generated by Java Sever Page), and J2ME (Java 2 Micro Edition). In the first prototype, users send SMS to interact with the system.
- The Model (runs the business logic of the application): the Model is composed of several Java interfaces and classes that implement these interfaces. The interfaces provide standard ways to access the business logic such as: searching free rooms, booking a room, etc. For example, if the developer finds a better algorithm to search the room (the result is a faster response time), only the class that implements the “SearchRoom” interface has to be changed. Other part of the system that use the Search function remain the same. We use the XML database Xindice to manage persistent data (hotel information, booking detail, etc). While it is possible to access the XML database from the Model classes, we did not use this approach. The reason is that it would require a lot of changes if we later want to change the database to relational database (e.g.: Oracle, SQL Server, etc). Therefore, we implemented a JavaBean that take care of database access. Thus, all classes use the database Bean to access the XML database. If we want to change the database server, only the Bean component need to be reimplemented.
- The Controller (coordinates the system activities, for examples: it receives the requests from the View and forwards the request to an appropriate Model component to process the request). Normally, in the MVC model, the clients (View) communicate directly with the Controller. In our system, there is a slight change: the clients firstly send requests to the SMS gateway and the gateway forwards the requests to the controller. This is because the short message system and the Web are different systems, they can not communicate with each other without the gateway. However, the SMS gateway is only required when we use SMS. Our system also support other client types such as J2ME and normal web browsers (the View are JSP pages). In these cases, the clients send HTTP requests directly to the Controller.
SMS gateway
Kannel is an Open Source SMS Gateway. It is used to access the easyHotel application (run on a Web server) through the GSM Network. A mobile phone is connected to a PC that runs the Kannel Server which manages incoming and outgoing SMS. Incoming SMS are searched for predefined patterns (keywords). If a match is found, the Kannel sends a HTTP GET request to the easyHotel servlet running inside Tomcat, the servlet engine. Outgoing SMS are created by sending an HTTP request from the easyHotel servlet to the Kannel Server.

Web server and servlet engine
Tomcat is a servlet container which runs the easyHotel Controller servlets and the database-access servlet. It is implemented in Java which provides platform independence and system scalability.

XML database
The Xindice database is used for managing persistent data in easyHotel. It is an open source JAVA implementation of the XML:DB API. It can be run as a servlet on the same servlet container as the easyHotel servlet or as a stand alone server. The easyHotel servlets and the Xindice server exchange XML encoded data by XML-RPC protocol (HTTP is used as transport layer). We use the Java-to-XML library named Castor for binding JAVA objects to XML data and vice versa.

4. Conclusions
The popular use of mobile devices in our daily life implies that M-commerce applications will be very common in the near future. In general, thin client approach is suitable for mobile devices as they have limited processing power. Testing M-commerce systems as well as E-commerce system can get benefit from the use of automated testing tools. However, we should not forget to test with the real devices because there might be some problems that can not be discovered by testing software.

References