

Mobile commerce for financial services - killer applications or dead end?

Michael Semrau and Achim Kraiss
Dresdner Bank AG
Corporate Center Information Technology
Enabling Technology Group
Research and Innovations
Juergen-Ponto-Platz 1
D-60301 Frankfurt am Main, Germany
(michael.semrau, achim.kraiss)@dresdner-bank.com

ABSTRACT

Since mobile commerce (m-commerce) started to be intensively discussed in the press, financial service companies are said to be the winners of m-commerce. But looking at existing m-commerce applications, you will find "really" interesting information only on few sites. In addition, there are many of these applications, which are just in a prototype state and not yet available to the customers. Based on the lessons we have learned from building prototype and productive m-commerce applications, the following paper briefly discusses some of the problems to be solved by corporations that want to use emerging innovative mobile technologies, such as the Wireless Application Protocol (WAP). In particular, we address issues regarding application design and development, security, and infrastructure.

INTRODUCTION

Times where m-commerce was seen as a gold mine have gone. Facing the real problems, the high expectations have been changed into a more realistic view on what is possible and what is not. On the other hand, it is still highly attractive to offer a new access "channel" to financial products for mobile customers. In addition to conventional access channels to financial services, such as branch office, the Internet or a call center, mobile devices potentially enable millions of users to get access to financial products and services. However, before one can start with the development of mobile applications it is necessary to analyze the real needs of mobile customers.

We found that three basic questions have to be answered for every application, before a mobile version of the application will be built.

- Is it of real benefit for the customer to have access to the application at any time and from any location?
- Can the handling of the application be reduced to a simple user interface with few simple interactions?
- Is it possible to personalize the required information with respect to the limitations of the output device and with still maintaining the basic functionality of the application?

Considering these questions, we defined and realized a brokerage application last year that enables the customer to buy and sell stocks, to sign stock issues, and to get additional information about the stock market.

REQUIREMENTS ON APPLICATION DESIGN

M-commerce applications cannot be built by simply extending an existing e-commerce application by mobile clients. However, in many cases m-commerce applications are planned as an extension of existing e-commerce applications and it is difficult to identify general rules to be applied when an e-commerce application has to become mobile. Our experiences have shown that every part of the application has to be checked for the specific concerns of mobile applications. The following example illustrates this:

In Germany the presentation of information for the signing of a stock issue underlies certain legal requirements. In an e-commerce application with front-ends like PC's this requires the display of a longer disclaimer with legal information. A similar presentation on a mobile device like a mobile phone or a personal digital assistant is not acceptable for the customer as reading the disclaimer may take the user multiple interactions and thus may take minutes. Together with the legal department we had to find another solution, which is acceptable for mobile users on one hand and fulfils the legal requirements on the other hand.

This example shows that m-commerce applications require specific mindfulness not only from an information technology point of view but also from a business perspective. We have identified the following critical issues for the acceptance of m-commerce applications.

- *Input should be restricted to a minimum.* On mobile devices, in particular on mobile phones, complexity of required input is one of the most critical factors for the acceptance of the application. It is important to avoid the input of alphanumeric data as far as possible. If the user would have to enter a lot of alphanumeric data for specific functionality you should rather drop this functionality. In general, complex input should be deferred to the latest possible point in time within the process when it is clear to the user that he requires to

input the data in order to get the requested information. For example, the user should enter his name for authentication only if authentication is really required, i.e. if private information has to be accessed. In many existing e-commerce applications, authentication has to be done at the beginning of a session, regardless of whether the user wants to access private or public information.

- *Focus on really important information.* Even if new mobile radio standards, such as General Packet Radio Services (GPRS), promise a substantial increase of bandwidth, the amount of usefully displayed information is restricted because of the limitations of the mobile device. That is why it is absolutely necessary to customize the information to the needs of the user. The important question which has to be considered for m-commerce applications is which information which user in which step of the process needs.

APPLICATION DEVELOPMENT

In this section, we discuss selected problems and solutions for the efficient development of WAP applications, i.e. mobile applications based on the Wireless Application Protocol (WAP). To address this issue in a more general context, we have conducted a research project in the IT Research and Innovations department of the Enabling Technology Group (enateg) of Dresdner Bank AG. The research project was motivated by a series of problems which came up when we developed and evaluated the first WAP prototype application in the bank.

Problems

In this section we briefly discuss the key problems we have identified for efficient WAP application development:

- *Lack of sophisticated toolkits.* While there are highly sophisticated development toolkits for conventional programming languages, script languages, and HTML pages, there are no such toolkits for the convenient development of WAP applications generating WML pages.
- *Incompatibility between toolkits and WAP phones.* Existing toolkits focus on the "simulation" of WAP phone display capabilities. However, we quickly found out that the display capabilities of the simulated phones provided by the toolkits and the "real" WAP phones are highly different, even if the toolkit and the phone come from the same manufacturer.
- *Heterogeneous display properties.* Each WAP phone has specific functionality, display properties, and each built-in WAP phone browser shows specific display behavior. To give an example, specific WAP browsers insert a "new line" after each link while others do not. Even different releases of WAP browsers from the same manufacturer, such as Nokia to name only one of them, show different display behavior. Hence, the

output of an application may completely differ from device to device and all these differences have to be addressed by the WAP application.

Our experiences led us to the following insights:

- Existing WAP toolkits are not sufficient for WAP application development at all. They can be used only for initial proof-of-concept rather than for productive applications. For productive applications, all tests have to be done with real target WAP devices.
- It is not realistic to let the application developer integrate WAP device adaptation functionality into the application, i.e. the functionality of adapting WML pages to a wide spectrum of different WAP devices. Considering multiple WAP devices in an application would lead to highly increased code complexity and development costs. Adaptation to different WAP devices should be addressed by an application-independent transcoding framework which we discuss next.

Solutions

In our research project, we have developed "best practices" in terms of guidelines for the efficient development of WAP applications. These guidelines give recommendations how WML pages should be structured when they have to be displayed at many different WAP devices. For example, the guidelines recommend that tables should not be used as many browsers do not display tables properly, and that images (wireless bitmap files) have to be inverted before being displayed at specific WAP phones as some WAP phones always invert images themselves.

Our experimental results based on the evaluation of many different WAP devices led us to the implementation of two different transcoding frameworks. Each transcoding framework enables the WAP application developer to ignore the specifics of different devices to the largest possible extent. The developer can concentrate on the generation of standard WML, and the transcoding framework transforms the standard WML code into device-specific WML code at runtime. Using a transcoding framework avoids "spaghetti-code" in the application, which makes the application code easier to maintain and consequently leading to reduced development time and costs. The transcoding framework basically consists of the following functionality:

- *Adaptation of image size.* The framework scales images to the specific display size of the target WAP device.
- *Image inversion.* The framework inverts images when they have to be displayed at WAP devices which always display inverted images.
- *Alternative realizations of non-implemented commands.* Many WAP browsers have not yet realized commands such as BOLD, ITALIC, etc. The

transcoding framework provides alternative implementations for these commands. For instance, the pop-out effect of BOLD letters is alternatively realized by the display of capital letters.

- *Realization of homogeneous layout.* The transcoding framework ensures- to the largest possible extent- a homogenous layout of displayed WML pages. For example, it inserts "new line" commands after specific page elements as some WAP browsers inherently do this as well.
- *Compensation of bugs.* The transcoding framework can even compensate bugs of specific WAP browsers. For example, we have found WAP browsers which generally decrement the size of input fields. To compensate this, the transcoding framework generally increments the size of input fields.

The two transcoding frameworks we have developed are different in their underlying technology. The first framework is suitable for all WAP applications where the presentation layer, i.e. the "view" in a model-view-controller architecture, is realized by Java servlets. In this case, the transcoding framework consists of a Java class library providing methods for the generation of WML output (see figure 1).

The WAP application calls the methods of a generic *Standard Device* class for the generation of a WML page, such as *startOfPage* for generating the start of the page, *setBold* for displaying subsequently generated text with bold letters, *printText* to display text, *writeLink* for displaying a link, etc. A global function *WapDevice.GetDevice(..)* is initially called by the application and returns an instance of a specialization of the *Standard Device* class (e.g. an instance of the *Nokia7110* class for the Nokia 7110 phone), depending on the specific WAP device where the WML page has to be displayed. This specialized class generates WML code which is specific to the corresponding WAP browser. The WAP device is recognized by the framework based on the *User-Agent String* of the WAP browser which is found in the header of the page request and which has to be passed to the transcoding framework.

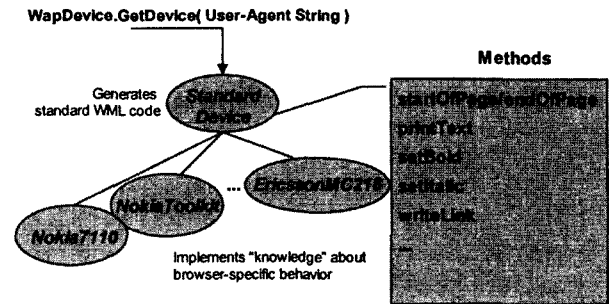


Figure 1. Java-Class-based Transcoding Framework

The Java-Class-based transcoding framework was able to keep the specifics of WAP devices completely transparent to the application developer. The usage of the framework within our WAP application has shown that the framework is easy to use as only Java skills are required and that the framework shows good performance. The major drawback of this solution is that its usage is restricted to Java Servlets. In many applications with a "cleaner" separation of content, logic and presentation Java Server Pages (JSP) are used for the presentation of WML pages. Therefore, we have developed a second transcoding framework which can be used for all sorts of Web-based applications producing WML output.

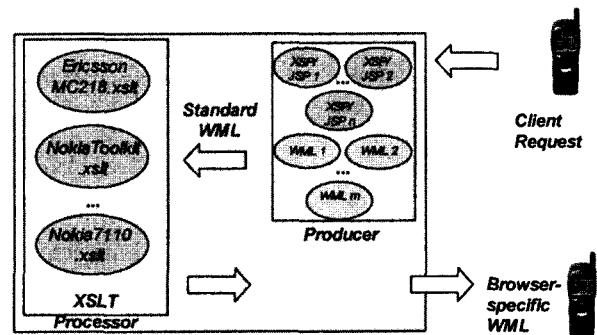


Figure 2. XSLT-based Transcoding Framework

The second transcoding framework we have developed is based on XSLT stylesheets (see figure 2). It is assumed that the WAP application produces standard WML output. Once a client requests a WML page, this standard WML output can be generated by static WML pages or dynamically by the execution of Java Server Pages (JSP) or Apache's Extensible Server Pages (XSP). The WML output of the application is piped into an XSLT processor where an XSLT stylesheet is applied for transforming the WML output of the application into device-specific WML for the target device. We have taken Apache's Cocoon

Framework and IBM's WebSphere Transcoding Publisher as "engine" for the execution of the XSLT stylesheets. In both cases, the major functionality of these engines is to select the appropriate stylesheet based on the User Agent String of the requesting WAP device and to apply the selected stylesheet to process the transformation of the application's standard WML output into the browser-specific WML to be passed to the WAP device.

The XSLT-based Transcoding Framework allows the "clean" separation of content, logic, and presentation in the application. Regardless of the technologies the application uses (Servlets, JSP, XSP,...), the application simply produces standard WML code and an explicit transformation of the application's output realizes the transcoding into device-specific WML. As compared to the Java-based transcoding framework, the explicit transcoding step leads to slight performance degradations of the XSLT-based transcoding framework. Furthermore, we have found that XSLT stylesheets are harder to maintain, especially when they become more complex or contain transformations outside the scope of "tree modifications". For instance, with XSLT stylesheets there is no elegant way to make a simple transformation of a text line into capital letters.

INFRASTRUCTURE - THE EXPENSIVE PART

Most of the WAP sites, which are offering information, are accessible via arbitrary WAP gateways. If personal data is transmitted or transactions are executed, a higher security level is needed. For this reason, WAP applications provided by banks are in most cases exclusively accessible via dedicated gateways installed in the bank. This additional effort and costs is caused by the missing end-to-end security of a WAP connection that cannot be achieved with existing hard- and software. From the viewpoint of financial services, the lack of end-to-end security can be seen as one of the major handicaps for the emergence of WAP technology.

We have realized an internal WAP infrastructure for the brokerage application that has been already mentioned in

the first section of this paper. Once the conception phase was completed, the realization of the infrastructure was a relatively straightforward process. However, the real problems were found in the details. We have identified incompatibilities between the gateway and some end user devices. In some cases, the mobile phone was not able to establish a WAP connection with the gateway. Furthermore, the realization of our security concept was awkward because of problems with digital certificates. We also found that existing WAP gateways cannot be integrated into existing systems management processes and software very well.

CONCLUSION

Our experiences with mobile applications and their infrastructure show that the realization of m-commerce applications, which give added value to users, is still expensive due to the application design and infrastructure issues discussed in this paper. However, the realization of mobile applications is worth the efforts, because this way we get optimized mobile applications which are tailored to the specific needs of mobile users.

In the future we will extend our research activities towards a transcoding approach that is more general in the sense that many different device types can be used to get access to a single application (sometimes also known as "write once, publish anywhere"). Assuming that the application generates XML documents, we will investigate how these documents must be enriched with information and how transcoding technologies can be applied to present content in HTML on desktop PCs, in PDF on a fax machine, in WML on a PDA, in WML on a mobile phone, etc.

In addition, we will investigate the impact of new mobile radio standards, such as GPRS and UMTS, on banking applications, as we strongly believe that the emergence of these standards with new cost models and increased bandwidth will significantly increase the potential for the widespread use of m-commerce applications.