AN INTEGRATED APPROACH TO DEVELOP PERVASIVE MOBILE APPLICATIONS

Hiata Anderson de O. Coelho, Ricardo de O. Anido and Alisson Luiz da Cruz
Institute of Computation, State University of Campinas, Av. Albert Einstein, 1251, Campinas-SP, Brazil

Keywords: Pervasive Computing, Mobile Application, Framework.

Abstract: This article introduces the QuickFrame, a development tool that allows mobile applications to run in several different types of mobile devices. That is only possible due to the fact that the QuickFrame possesses a standard specification language and has the capability of defining the application interface, verifying the target devices' specifications and pre-visualizing the user interface. Therefore, several of the problems caused by the large number of mobile device models available and their different specifications are eliminated. Thru reaching to that goal we have a tool to make mobile applications more pervasive.

1 INTRODUCTION

Many people have become familiar with the use of devices such as mobile phones, PDAs, etc. This is a great opportunity in terms of providing a great mass of customers with mobile devices and services. Besides, more and more tasks that need phone operators are being replaced by automatic services that receive messages and carry out suitable actions without any human intervention. This trend will continue to increase, since the economic benefits provided are enormous.

Device and mobile computational technologies have changed very rapidly in the past few years. With that happening, we now have the existence of a huge number of different mobile devices with different technologies sharing the same market. Those facts impose, in a very paradoxical way, a great limitation in the expansion of mobile systems software development due to the vast heterogeneity of those devices.

Heterogeneity is an intrinsic characteristic of pervasive environments (e.g. mobile phones). Plug-and-play facilities must be provided for such environments to allow clients from different locations to access remote services through various devices. These devices have different functionalities and do not necessarily implement all the standards of the requested service, such as communication protocols, information visualization mechanisms and so on. Therefore, pervasive environments are designed to facilitate adaptability and run-time reconfiguration (Mostefaoui et al., 2004).

Pervasive computing (Weiser, 1991) is a computing era with a special distinguishing characteristic that users will no longer be tied to the desktop paradigm, and will therefore become increasingly mobile. Hence, this will result in establishing new patterns that are quite different from what we have known traditionally as workflow or office work. These new usage patterns will be performed by a large number of low-power devices (mobile phones, PDAs, palmtops), co-existing with desktop computing systems, disconnected operations, and rapid and ad-hoc changes in usage patterns (Bellur and Narendra, 2005). In the case of pervasive computing, systems changes in usage patterns are the norm, therefore, fast deployment of mobile application is essential.

However, there is an obstacle that must be transposed in order to facilitate the development of pervasive mobile applications. That obstacle is the absence of tools to make development more adjustable to the devices' different models, capacities and technologies.

This work address to this problem proposing a framework based in open patterns to describe mo-

Anderson de O. Coelho H., de O. Anido R. and Luiz da Cruz A. (2007).
AN INTEGRATED APPROACH TO DEVELOP PERVASIVE MOBILE APPLICATIONS.
In *Proceedings of the Second International Conference on Wireless Information Networks and Systems*, pages 181-188
DOI: 10.5220/0002148501810188

bile applications as follows. Section 2 summarizes our motivation. Section 3 shows the related work. Section 4 presents the highlights of the QuickFrame architecture, and gives the details of XForms, interpreters, QuickFrame Designer and Server. Section 5 gives our conclusions and section 6 future work.

2 MOTIVATION

Mobile devices have several limitations when compared to desktop PCs and laptops (Passani, 2007).

- Small screens: Mobile devices have small screens. Even today, the average mobile device allows for 20/25 character per line and 5/7 lines of visible text. In these conditions, simply scaling down a application to fit a mobile device is bound to be catastrophic in terms of user experience.
- Limited input capabilities: The majority of mobile devices are phones with a numeric keypad. While this is sufficient for dialing phone numbers, entering text with it is a time consuming.
- Limited processor power and memory: Mobile devices have limited processor power and little memory compared to desktop or even server platforms. In order to be successful, your application needs to make most of the available resources.
- Limited bandwidth: Mobile devices have little bandwidth available when compared with PCs on the Internet. With the advent of 3G (EDGE, UMTS, HSDPA) the situation has improved for some users, but a lot of other users (or 3G users themselves when roaming out of 3G coverage) can only count on a speed of just a few kilobytes per second (GSM, GPRS). Latency introduced by the time needed to establish a connection should also be accounted for.

In addition, there are also relevant differences in the value users assign to services depending on how critical those services are. Longer access times and higher costs are factors that discourage users of a given web site or application from accessing its mobile counterpart. These limitations have serious implications on the way one should design mobile applications.

Porting mobile applications generally requires developers to adapt themselves not just to the several limitations listed above, as to the differences in the screen resolution, processor speed, memory thresholds, and sound capabilities, all of which can vary widely from device to device. For publishers, this can not only exponentially increase application development and asset creation time, but can also cause them to miss critical time-to-market window in a hypercompetitive industry (Jaokar, 2006).

In order to facilitate the creation of mobile applications and make them more pervasive in a real word (e.g., using mobile phones) it is necessary to transpose the so called Device Diversity issue, as shown in figure 1. Device diversity is the problem of having to write separate or custom software code for each device on which your application must run. In other words, the problem consists in writing and managing a separate code for the same basic functionality just to meet individual device capabilities and/or features. The worst-case scenario is when developers are forced to write a separate code for each and every device they want to support. Device diversity turns the idea of creating and mass distributing an application into an almost insurmountable problem.

Figure 1: Device Diversity.

This article presents a solution for the Device Diversity problem, using XML based descriptors for mobile applications and interpreters for these descriptors. The adopted XML format is the XForms (Pemberton and Boyer, 2006), a standard based in the World Wide Web Consortium (W3C). W3C is an international consortium where member organizations and the public work together to develop Web standards.

3 RELATED WORK

The specific requirements of pervasive applications have been widely discussed in (Banavar et al., 2000; Banavar and Bernstein, 2002). In particular, Banavar et al. describe a programming model that strictly treats task logic and user interaction separately. They make the suggestion to start with creating a superior task-based model for program structure that covers the user's abstract interaction and the application logic, and then continue with creating a subordinate

navigation model that covers the flow of the view elements.

There are some pervasive computing projects that aim at devising a high-level UI design language for abstract user interaction (Patern and Santoro, 2002; Giannetti, 2002). Any of these approaches proceed similarly: the modeling phase of the abstract user interaction in the respective language is followed by a semi-automatic generation of the device-specific code. Although our paper describes a similar automated development process, in the QuickFrame project, application descriptors are generated to be interpreted in mobile devices, as shown in the subsequent sections.

Proposed by (Fernandez et al., 2005) MoviForms is a system that generates forms-based services for mobile phones. It has been developed using the following technologies: J2ME (Java 2 Micro Edition) and SMS (Short Message Service). MoviForms is a very interesting solution, but the overals solution limits the use of XForms to SMS dependent devices.

So far, IBM, Oracle and FormFaces have produced significant initiatives to cope with the use of XForms in mobile devices. IBM Forms for Mobile Devices (IBM, 2004) is an implementation of XForms that runs on Palm-powered devices and shows how XForms can be used to create form-based enterprise applications running on mobile devices. This package has been released by IBM on IBM alphaWorks, the company site for showcasing emerging technologies. On March 2004, Oracle Wireless Client (Oracle, 2004) has released a preview of the Wireless Client. Despite the name, the software does not run on a mobile device, but contains a Web browser plug-in (hence maybe the preview in the name). The plug-in runs XForms on the client device, just like a browser would if it supported XForms natively. FormFaces Mobile Solution (FormFaces, 2007) is a JavaScript implementation targeted to run within any DOM Level 2 compatible browser, it supports many OS deployments including Palm and Pocket PC mobile devices. Although they are excellent projects, they are not focused on device diversity problem.

4 THE ARCHITECTURE

The purpose of the QuickFrame project is to develop tools to minimize the Device Diversity problem, making mobile applications more pervasive. The solution found in the project was to create mechanisms for describing and executing mobile applications in a way that those applications can run in a great variety of different devices. QuickFrame's architecture is divided in three parts. Each part has it's own well defined responsibilities and communicate with the others thru XML. The parts are:

- QuickFrame Designer (QFDesigner) - An Eclipse plug-in (Eclipse, 2007) used to describe screens and flows in mobile applications.
- QuickFrame Interpreter (QFInterpreter) - Runs in the mobile devices and has the responsibility of interpreting applications created in the QFDesigner, that is, generate forms automatically and guarantee their execution.
- QuickFrame Server (QFServer) - Responsible for storing applications described in the QFDesigner, exchanging information with mobile applications running in the interpreter and interchanging information with storage systems (database) and/or legacy systems.

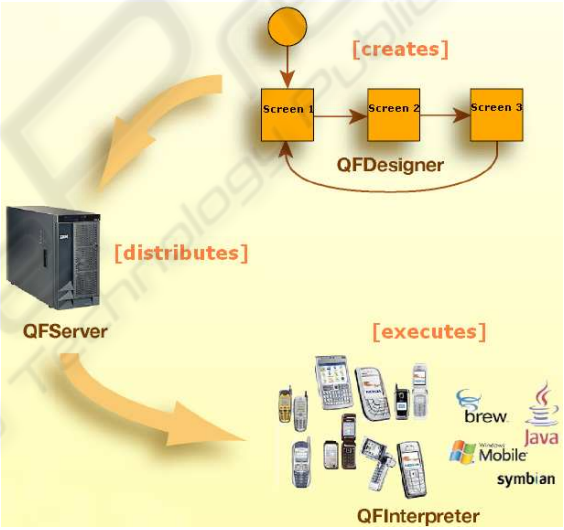

Figure 2: QuickFrame Architecture.

Each of the parts described in the picture 2 will be better described in the next sections. The link between the three parts is the XForms standard in the XML format, as described in the next session.

4.1 XForms

XForms (Pemberton and Boyer, 2006), (Boyer et al., 2006) is an XML format used for specifying user interfaces, specifically Web forms. In fact, XForms was designed by W3C to be the next generation of HTML/XHTML forms, but it is generic enough so that it can also be used in a standalone manner to describe any user interface, and even perform simple and common data manipulation tasks.

By default, an XForms document is composed of two main modules which are called XForms Model

and XForms User Interface. These modules are directly usable inside other XML formats, thus yielding different presentation options. The XForms Model is basically an XML Schema or XML Instance for the data managed by the document. It describes the structure of the instance data, allowing client-side verification, and separating data from application logic and presentation. All information exchange is done through XML documents.

The XForms User Interface describes the interaction with the user through abstract controls. By abstract one should understand that the controls define only the intention and not specific types of user interface elements. By using this approach to describe the user interface, XForms becomes device and platform independent, since the concrete user interface elements will be instantiated by the XForms processor accordingly to the device and/or platform in which it is running.

The Xforms was chosen as the standard for description after analyzing and comparing it with other standards in the recent literature. Those other standards include XAML (Griffiths, 2004) e UIML (Abrams and Helms, 2004). However, the presence of a data model and the persistency of this data in a single document are very attractive features of the XForms standard for the QuickFrame project. Those features collaborate in the information exchange between mobile devices, (which can run online or offline) and servers.

Considering that one of the goals of this work is to provide an easy manner to treat device and platform fragmentation in the mobile environment, XForms was selected to be the language for user interface description. Other known benefits of XForms for mobile devices are:

- Data, logic and presentation separation.
- Reduction of the amount of work required to target multiple devices/platforms, since it allows user interfaces to be described abstractly.
- More self-contained, because it requires fewer round trips to the server.
- Ability to suspend and resume the completion of the form data, which is very important in the context of sparse network availability.
- Reduction of the need for JavaScript, more advanced controls and data type verification, which is particularly interesting as JavaScript support varies greatly on mobile devices and cannot be widely relied on.

4.2 QuickFrame Designer

The QuickFrame Designer (figure 3) is an easy-to-use set of graphical tools and interfaces for expediting the creation and editing of documents with XForms-based content. It is an Eclipse visual plug-in, which makes the process of creating documents with XForms content easier, faster, and simpler. Its main objective is to provide a fast way to create a specification for a mobile application , through which screens and application execution flow can be defined. At the end of the specification step, an XML file with all the information on screens, screen elements and the flow (screen navigation) is generated.

The XForms format file will be used for the automatic screen generation in the devices, as we will see soon. Some of the capabilities of the Designer are:

- A visual, palette-driven editor.
- A source view of the XForms document.
- Integration with the standard Eclipse Properties, Outline, and Problems views.
- An XML Instance view.
- Interfaces for easily binding XML instance data to XForms constructs.
- The ability to view or update XForms attributes.
- Mobile application preview.

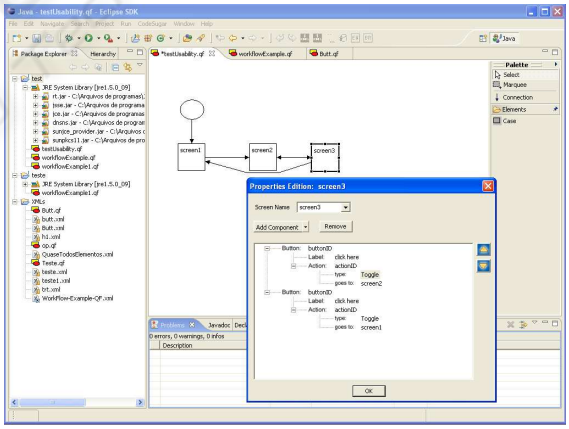

Figure 3: QuickFrame Architecture.

The QuickFrame Designer is tightly integrated with Eclipse and includes convenient user interface elements such as editors, context menus, wizards, and other XForms and XML related views. Furthermore, this product provides constructs for accelerating XForms-based design, such as type constraints, drop-down lists, text input, tree-based interfaces, and other graphical interface aides. From these powerful graphical interfaces, the XForms document is serialized to an XML document that is built

to include XForms elements and attributes compliant to the XForms 1.0 standard. The edited XForms document can immediately be rendered in popular XForms-capable rendering tools.

Several XForms defined screen elements can be inserted on the applications' forms, such as the Button, Input, Output, TextArea, Secret, Select and Label element. It is important to notice that these components correspond to the XForms elements that are relevant for applications in mobile devices according to our own analysis and needs. According to the XForms specification, each element has a set of obligatory attributes which must be filled in with data. To edit each element's values and attributes, the user simply needs to click the desired element and edit its value. Some attributes won't allow text input, instead, they will allow choosing among a set or range of choices. Once all of those restrictions and specifications are put together, it is possible to guarantee the application's consistency and correctness when translated to the XForms format.

Once all the necessary elements in each one of the application's screens have been set, the XForms document can be generated for later interpretation in the mobile devices. Another functionality of the QFDesigner is the export of the whole project in an XML document, thus saving the status of each object and element on the diagram. This facilitates information interchange between development teams, and, in the future, that concept can be improved so that editing the XML document will be interpreted by the QFDesigner as changes in the project's diagram.

There is another important feature of the QFDesigner which is the pre-visualization of the built application. Through that function, a preview of the positioning and appearance of the screen elements is possible before the application is sent the mobile devices.

For each of the application's screen, a data model is created. That data model is formatted and inserted in the XForms document as an instance data. This way, it is possible to have an explicit separation between data and user interface during the execution on the mobile device.

4.3 QuickFrame Interpreter

In order to facilitate data communication over mobile networks, content providers need to serve contents that are presentable and functional on mobile devices. However, there are some challenges related to the inherent nature of mobile communication, as compared to the traditional wired Internet.

The challenges for content providers to present contents to mobile devices lie in the physical characteristics of mobile devices and network. There is an enormous variety of mobile devices in the market, and each has different hardware and software configurations, such as screen size, graphic capabilities, input mechanisms, processing power, memory, operating systems, micro-browsers, protocol support, etc. Data transfer rates on a mobile network, which has originally been designed for voice transmissions, are low (at least before 3G services are fully in effect) and the transmission is unreliable. Essentially, content providers need to serve different contents to mobile devices with different hardware, software interface constraints and network configurations.

The interpreters are software solutions that, through the use of XForms, enable pervasive mobile devices to access and complete form based applications. This solution allows developers to quickly create, deploy, and use form based applications. The interpreters can also deal with the problem of intermittently connected mobile devices that need to access and complete business forms stored locally on the device. The completed forms are transferred to a server for additional processing when connectivity is available. Forms are pushed to mobile device clients and cached on them. As the mobile device user completes forms and submits them, the completed form instances are queued for delivery to the Server. When the Server receives the completed form instances, it dispatches them to the target backend application to finish their processing or send a response to the device.

These interpreters belong to the mobile application execution division in the QuickFrame Project, which means, the interpretation of the applications' description.

The interpreters automatically create screens and locate the screen elements in the correct way, that is, they transform XForms files, as shown in Figure 8, into mobile applications, as shown in Figures: 4, 5, 6 and 7.

Currently there are many interpreters in various technologies being developed: Java Platform Micro Edition (JME), Windows Mobile, BlackBerry and Wireless Application Protocol (WAP). These interpreters run in mobile devices, reading the XML document in an XForms format and dynamically generating the application's screen, this way, applications are described only once and can be executed (or interpreted) in several devices and platforms.

These interpreters have a basic set of functionalities:

- Access the server to obtain the description of the application (XForms format file).
- Interpret the XForms file and automatically gen-

erate application's screens

- Guarantee that the application's execution flow occurs the same way it was specified by the QFDesigner.
- Exchange information with the server during execution time.
- Guarantee that screens are smartly created, which means, correct positioning of the elements, respecting capacity and display size of each device.
- Operate on-line and off-line.

Figure 4: QFInterpreter on SonyEricsson JME Simulator phone.

Figure 5: QFInterpreter on BlackBerry Simulator Phone.

Figure 6: QFInterpreter on Nokia JME Simulator phone.

Figure 7: QFInterpreter on PocketPC Simulator Mobile Windows.

4.4 QuickFrame Server

Aiming at a more simplified way of supplying mobile applications and the integration guarantee between legacy systems and mobile devices, enters in scene the serving side of the project, called QFServer. Once the application was described in the QFDesigner it is sent in the XForms format to be stored in the server. To better understand the role of the server in the QuickFrame architecture consider the following XForms archive (figure 8).

When the application arrives at the QFServer it is stored in the database and will be available to be downloaded by the mobile devices. When the QFInterpreters request the applications, the QFServer search all available applications for that user and return a XML that contains a list of applications.

Then, the QFInterpreter receives the XML and shows the same list for the user to choose an application to download in the device. The selected application is downloaded and stored in the mobile device for future execution. On the server side, after the receiving and storing of the application, the QFServer makes the syntactic analysis of the XForms file, more specifically on the Data Model part, as detached in the figure 8.

The Data Model represents the information contained in the XForms and informs which format will be used for the information exchange between server and mobile device. After the syntactic analysis, the QFServer constructs objects that represents the information contained in the Data Model, as shown in the figure 9.

An application can contain more than one data instance in your Data Model. All this process is done in order to make possible the information exchange between server and device at the exact moment of the application execution on the mobile device. That is, in determined moment of the application execution the device will request some informations. In the case of the Data Model shown in the figure 8, the QFServer will answer, for example, with the XML shown in the figure 10.

```
<xforms:model>
    <xforms:instance id="screen1">
        <screen1>
            <Name/>
            <Phone/>
        </screen1>
    </xforms:instance>
</xforms:model>

<xforms:switch id="main">
    <xforms:case id="screen1">
        <xforms:input ref="Name">
            <xforms:label>Name</xforms:label>
        </xforms:input>
        <xforms:input ref="Phone">
            <xforms:label>Phone</xforms:label>
        </xforms:input>
        <xforms:trigger id="buttonID">
            <xforms:label>Ok</xforms:label>
            <xforms:action ev:event="DOMActivate">
                <xforms:toggle case="screen2" />
            </xforms:action>
        </xforms:trigger>
    </xforms:case>
</xforms:switch>
```

Data Model

Figure 8: XForms Sample.

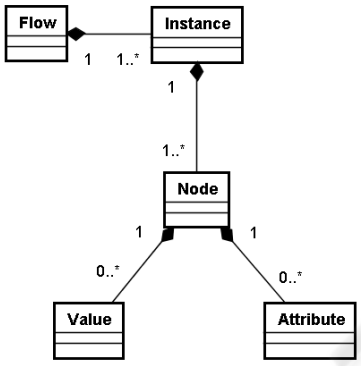

Figure 9: Object Diagram.

```
<screen1>
    <Nome>XYZ XPTO</Name>
    <Phone>12345678</Phone>
</screen1>
```

Figure 10: Instance Data.

The same is true for the information sent from device to server. The Data Model is filled in the device and the server keeps the information in the objects Node/Value and Node/Attribute. Later these objects are stored in a persistent way in the database.

To summarize, the main functions of the server are:

- To receive and store the application created in the QFDesigner.
- To supply applications for the QFInterpreters.
- To transform Data Model into objects.
- To change information with the QFInterpreters through the Data Model.
- To allow that the QFInterpreters create, read, update and delete informations in the Data Models.

5 CONCLUSION

Running a mobile application in different devices regardless model or technology is a recent challenge, because, as systems and devices evolve, they become more complex. The competition among manufacturers increases the variety of device specifications. This forces the companies that develop mobile applications to supply a larger number of services for the population that uses applications in their mobile devices. Such diversity combined with the expectation of the population that benefits from mobile computing, makes the development of mobile applications more complex. This complexity is minimized with QuickFrame.

By creating application descriptor rather than application code, the same complex applications can be used for many different devices in different mobile operational systems. By creating the toolkit based on the same patterns described in XForms, developers will be able to easily create new visual representations for mobile applications. By creating interpreter for these application descriptors, we expect mobile applications to run in a large number of devices. By adding interpretation support to XForms files in devices, developers will be able to create more portable and pervasive applications through new and innovative techniques for describing and running mobile applications.

In order to attain the objectives of the project, there was the necessity of proposing generic models in all three parts of QuickFrame. In other words, the utilization of a W3C defined format (XForms) for describing the applications, mechanisms that were portable to several devices (interpreters) for processing that format and having generic data being treated on server side. All these parts working together guarantee the capacity for developing a wide range of types of mobile applications and also the ability to manipulate and exchange information on them.

6 FUTURE WORK

Two main future developments are being developed: a tool for verifying the consistency of the applications built using QuickFrame and a tool for previewing the screens generated by the interpreters in different devices. The constraints verifier will be a tool to detect inconsistencies in the interface specification for a specific device, such as screen resolution, input controls availability, memory limitations, etc. These inconsistencies often cause difficulties in browsing contents. This verification process will be implemented as a set

of validation rules, configured dynamically. It will generate log files that contain error and warning messages, so the user can use the information to help troubleshoot any issues that might occur.

After the forms design process, a user interface preview is essential to verify the applications behavior in each device. The QFDesigner already have the preview function, but we propose a simulator for a set of devices using specific screen generation rules. This tool will render the interface specification with specific device interpreters, using a subset of each device specific interpreter rendering code. It will allow the user to view how a mobile activity will look when it is displayed in a variety of different output devices, each of which with different capabilities. There are a number of problems that must be solved in each device implementation, so the interpreter has some intelligence on how to render the interface elements regarding the small display size and/or limited number of buttons of most mobile devices. For instance, an XForms document that has lots of triggers (typically rendered as buttons) could be rendered as a list of options with a select button, in accordance to the limited number of physical buttons present in a mobile device.

In order to generate a preview in the simulator closer to the reality, device specifications inserted by user and also the information contained in the WURFL base will be used. The Wireless Universal Resource File (Trasatti and Passani,) is an open source XML configuration file which aggregates the features and capabilities of a number of mobile devices. WURFL targets developers who need to programmatically obtain features and capabilities of mobile devices in order to develop wireless applications.

ACKNOWLEDGEMENTS

The authors and all members of QuickFrame project are grateful to FINEP, Intel Corporation and Compera for all kind of support received.

REFERENCES

Abrams, M. and Helms, J. (2004). User interface markup language (uiml) specification. Available at: http://www.uiml.org/ [Last accessed on March 2007].

Banavar, G., Beck, J., Munson, E. J., Sussman, J., and Zukowski, D. (2000). Challenges: an application model for pervasive computing. *Proceedings of the 6th annual ACM international conference on mobile computing and networking*.

Banavar, G. and Bernstein, A. (2002). Software infrastructure and design challenges for ubiquitous computing applications. *Communications ACM*, 12(45):92–96.

Bellur, U. and Narendra, N. (2005). Towards service orientation in pervasive computing systems. *International Conference on Information Technology: Coding and Computing.*, 2:289–295.

Boyer, J., Landwehr, D., Merrick, R., Raman, T., Dubink, M., and Klotz, L. (2006). Xforms 1.0 specification. Available at: http://www.w3.org/TR/xforms/. [Last accessed on March 2007].

Eclipse (2007). *Eclipse IDE*. Available at: http://www.eclipse.org/ [Last accessed on March 2007].

Fernandez, C., Pece, J., and Iglesia, D. (2005). Moviforms: Xforms for mobile phones.

FormFaces (2007). *FormFaces Mobile Solution*. Available at: http://www.formfaces.com/. [Last accessed on March 2007].

Giannetti, F. (2002). Device independence web application framework (diwaf). *Proceedings of the HP Labs W3C workshop on device independent authoring techniques.*, pages 25–26.

Griffiths, I. (2004). Inside xaml. Available at: http://www.ondotnet.com [Last accessed on March 2007].

IBM (2004). IBM: *IBM Forms for Mobile Devices*. Available at: http://www.alphaworks.ibm.com/tech/ifmd. [Last accessed on March 2007].

Jaokar, A. (2006). Ajax for mobile devices will be the hallmark of mobile web 2.0. Available at: http://linux.syscon.com/read/167026.htm. [Last accessed on March 2007].

Mostefaoui, G., Pasquier-Rocha, J., and Brezillon, P. (2004). Context-aware computing: a guide for the pervasive computing community. *The IEEE/ACS International Conference on Pervasive Services*, pages 39–48.

Oracle (2004). Oracle: *Oracle Wireless Client*. Available at: http://www.oracle.com [Last accessed on March 2007].

Passani, L. (2007). Global authoring practices for the mobile web. Available at: http://www.passani.it/gap. [Last accessed on March 2007].

Patern, F. and Santoro, C. (2002). One model, many interfaces. *Proceedings of the 4th international conference on computer-aided design of user interfaces CADUI.*, pages 143–154.

Pemberton, S. and Boyer, J. (2006). W3C, *XForms - The next generation of Web Forms*. Available at: http://www.w3.org/MarkUp/Forms/. [Last accessed on march 2007].

Trasatti, A. and Passani, L. Wireless universal resource file. Available at: http://wurfl.sourceforge.net/. [Last accessed on march 2007].

Weiser, M. (1991). The computer for the twenty-first century. *Scientific American*, pages 94–100.