Mabel Vazquez-Briseno *et al*, International Journal of Computer Science and Mobile Applications,
Vol.2 Issue. 5, May- 2014, pg. 20-35 **ISSN: 2321-8363**

Evaluation of Java-Based Platforms for mHealth Solutions

Mabel Vazquez-Briseno[1], Mariana Mendez-Flores, Elitania Jimenez-Garcia, J.I. Nieto-Hipolito

[1] Faculty of Engineering, Architecture and Design, UABC, Ensenada, Mexico
mabel.vazquez@uabc.edu.mx

Abstract

Mobile health (mHealth) has recently emerged as an important subsegment of electronic health. Particularly in developing countries, mHealth has the potential to improve access to healthcare services. An important factor for the adoption and sustained use of mHealth services is related to the availability and effectiveness of mHealth applications. This work focuses on exploring the features of mobile platforms that are especially suited for the development of mHealth applications and services. Results were obtained through an exploratory analysis, a quantitative experiment, and a qualitative evaluation with a focus group. These findings will provide valuable information about mHealth systems for developers, researchers, users, and decision makers. .

Keywords: mobile computing, software, health informatics, Java, Bluetooth.

1. Introduction

Mobile technology offers opportunities for preventing health problems worldwide. *Mobile health* or *mHealth* is the use of mobile communications and devices, such as mobile phones, for health services and information (Vital Wave Consulting, 2009). The term mHealth was first defined by Istepanian (Istepanian, 2004) in 2004, as "Mobile computing, medical sensors and communication technologies for healthcare". Several mHealth systems have since been deployed worldwide, such as those described in (Mougiakakou, 2009), (Masek, 2009), (Benlamri, 2010), (Moron, 2011) and (Marshall, 2009). A major challenge in the development of mHealth applications is that current mobile devices are very diverse, especially with regards to operating systems (OSs) and runtimes. This heterogeneity causes problems for developers, who must fit applications to as many devices as possible, including multiple versions of the same, in-use device. Moreover, the chosen platform must provide the features required by mobile applications intended to be used for health purposes.

There are several platforms that are currently available for mobile application programming. Among the most popular are: the iOS, Android, and Java Micro Edition (ME) platforms. The iOS platform, which is a proprietary platform for devices such as iPhone and iPad, offers state-of-the art technology. However, devices running iOS tend to be comparatively expensive. Android, which was developed by the Open Handset Alliance (OHA), and Java ME, developed by Sun MicroSystems, are Java-based platforms. Android is a free-source platform that allows the development of applications for devices running the Android OS, which includes a wide range of devices. Java ME offers isolation from specific mobile device hardware and specific OS software, by using virtual machine (VM) abstraction. For this reason, Java ME can be used in various devices, including "feature" and low-budget mobile phones, which are commonly used in developing countries. In this work, we explore the features of Android and Java ME that are especially suited to the development of mHealth applications in developing countries.

The rest of the paper is organized as follows: In Section 2 we present an overview of Android and Java ME platforms. In section 3, related works are presented. Section 4 describes and analyses mobile health systems. In

section 5 we present the evaluation metrics and case study. Section 6 presents the quantitative evaluation and obtained results. A qualitative evaluation using a focus group is presented in Section 7. We provide a series of recommendations in Section 8. Finally, Section 9 summarizes this paper.

2. Mobile Java-Based Platforms

Despite the overall heterogeneity of mobile device capabilities and features, countries show a marked tendency to use specific mobile platforms, as found in (Canalys, 2011) and shown in Figure 1. For instance, the iOS platform is preferred in countries, such as the United States, Canada, France, and the United Kingdom, whereas the Android platform is preferred in Mexico and South America. Most countries in Africa tend to prefer devices running Symbian OS and Nokia Series 40, both of which support Java ME as the development platform. Thus, in spite of increasing interest in so-called "smart phones," most of the world (and especially developing countries) continue to use "feature phones," which are less expensive and have limited resources compared to smart phones. Users in developed countries have access to the latest, more-expensive technology (e.g., iOS devices), whereas developing countries tend to prefer less-expensive devices (i.e., Java ME and Android).
Java ME and Android are very interesting platforms with a wide range of capabilities; both are based on Java, but they have very marked differences. In the next sections, we explore each platform and analyze their similarities and differences.

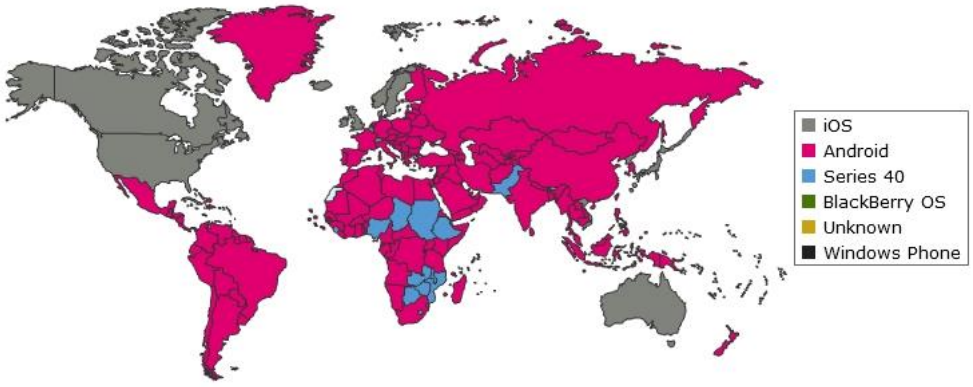


Figure 1. Top mobile operating systems per country (Source: http://gs.statcounter.com/#mobile_os-ww-quarterly-201402-201402-map)

2.1 Java ME

Java ME is a licensed platform that is a smaller version of Java 2 Standard Edition (also developed by Sun Microsystems). This edition was specifically developed to fit many devices, including feature phones, TV set-top boxes, e-readers, Blu-Ray readers, printers, and more, with a wide range of capabilities. Java ME is now managed by Oracle as part of the Java for Mobile Devices technology. It is used in many devices and OSs, such as Nokia's Series 40, the new Bada OS, and the Symbian OS together with native software. Implementations of Java ME are available for Windows CE, Windows Mobile, Maemo, and MeeGo.

The overall architecture of the Java ME platform can be described as follows (Helal, 2002):

• A VM targeted to the end-user (consumer) device. For mobile devices, the VM is called the kilo VM (KVM), where *kilo* refers to the small memory footprint.

• Libraries and application programming interfaces (APIs). These *Profiles and Configurations* allow the consumer to use the device's capabilities and other functionalities. They are grouped separately according to device type.

• Several tools to accompany development, deployment, and device configuration.

2.2 Android

Android is a complete software stack for mobile application development created for devices with constrained processing power, memory, and storage capacity. It was released and developed by OHA, particularly by Google. Unlike Java ME, Android provides a Linux-based open-source OS. The platform includes an application middleware layer, a set of APIs, and key application libraries, along with an SDK. As a result, heterogeneity is minimal among Android mobile phones, which have the same core applications, OS, and minimum hardware characteristics. Android permits developers to write applications that take full advantage of the mobile hardware. The Android OS is built on the open-source Linux kernel system. It uses a custom VM that was designed to optimize memory and hardware resources in a mobile environment. Android's VM, Dalvik, was developed by Google and differs from the VM used in Java ME (i.e., KVM). The design of Dalvik was intended to allow multiple VMs to run efficiently on the same device. To achieve this, the Linux kernel performs threading and low-level memory management. The Linux kernel also allows developers to write C/C++ applications that can run directly on the OS. Android also offers some unique features that facilitate the development of innovative applications (Meier, 2009).

2.3 Differences between Java ME and Android

Although both Java ME and Android are Java-based programming languages, they have some differences that justify their careful comparison when developing mHealth or other types of mobile applications and services:

- Java ME and Android rely on completely different VMs. Java ME uses KVM, which is similar to the standard Java VM but uses a smaller footprint. Android uses Dalvik, which was written to allow devices to run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik executable (.dex) format, which is optimized for a minimal memory footprint. This register-based VM runs classes compiled by a Java language compiler. Java classes are transformed into the .dex format by the included "dx" tool. The Dalvik VM relies on the Linux kernel for underlying functionality, such as threading and low-level memory management [Paul, 2010].
- Like Java Standard Edition, Java ME applications are packaged in Jar files. Applications installed "over the air" require a Jad file containing the manifest, which includes only a description of MIDlet Suite. Android applications are packaged into apk files. The manifest, Android Manifest is a mandatory file in all cases.
- Android allows the use of XML to implement the user interface. This option is not available in Java ME, in which all of the programming is done with Java.
- Android generates several resources and files, and strictly organizes projects. An auto-generated *R.java* file contains unique identifiers for various resources contained in the project (strings, layout, colors, figures, etc). Android allows the use of Android interface definition language (Aidl) files. The Android asset-packaging tool (aapt) compiles the application resource files, such as the *AndroidManifest.xml* and other XML files.

Figure 2 depicts development in the Android and Java ME platforms.

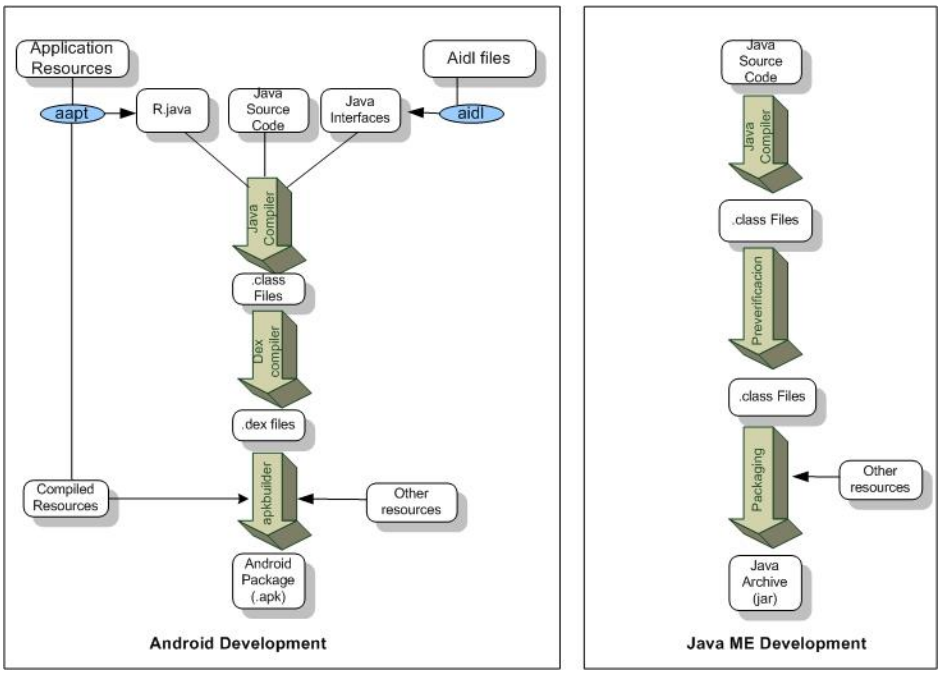


Figure 2: Development in Android and Java ME platforms

3. Related Work

In this work, we particularly considered mHealth applications. Previous studies have compared mobile platforms while targeting other kinds of applications and services. For instance, in (Oliver, 2009), Oliver presents the characteristics of mobile platforms necessary for the research of mobile networks. His work assesses 5 smart phone platforms: Android (Linux), BlackBerry, iPhone (Mac OS X), Symbian, and Windows Mobile. He describes the main features of each platform, to provide researchers with information for choosing a development mobile platform. He mainly analyses explicit features of these platforms, but does not present a performance analysis, which is the main difference from our work. In another study, Gavalas and Economou (Gavalas, 2011) compare Java ME, Android, .NET Compact Framework, and Flash Lite. Theirs is a very complete reference that provides insight about the main features of each platform, especially for the development of mobile games and multimedia support. They implement a stand-alone mobile game called Snake, identify some metrics, and emphasize differences between platforms. The analysed metrics include lines of codes, development effort, and deployment application size. Although we included some of these metrics in the present work, we emphasized features related to the development of mHealth applications, which strongly rely on the use of remote connection capabilities (e.g., network and sensor connections by Bluetooth). We provided information about these features in Java-based platforms. Other works, such as (Gronli, 2010), compare Java ME and Android in general, and not for a particular kind of application or service.

We present some results obtained with real devices and compare them to the emulator results. We also include a qualitative evaluation based on the use of the System Usability Scale (SUS) and a Computer System Usability Questionnaire (CSUQ).

4. Features of mHealth Applications

Particularly in developing countries where mobile phones are more prevalent than landlines and telephones, mHealth has the potential to improve access to healthcare services. According to Vital Wave Consulting (Vital Wave Consulting, 2009), mHealth applications in developing countries fall into the following categories: education and awareness, remote data collection, remote monitoring, communication and training for healthcare workers, disease and epidemic outbreak tracking, and diagnostic and treatment support. Several mHealth projects falling into these categories have been successfully deployed in developing countries, such as those described in (United Nations, 2008), (Nokia, 2010), (Feder, 2010), (Loughborough University, 2007), (Saran, 2009) and (West Health Institute, 2012). Another classification was proposed by Mirza et al. (Mirza, 2008) classifying mHealth into clinical and non-clinical applications. Clinical mHealth applications encompass the whole spectrum of healthcare, with services for prevention, remote monitoring, treatment, and patient support. Non-clinical mHealth applications are related to the smooth operation of health services and include administrative and management operations.
Despite the diversity of the mHealth services and applications, it is possible to identify some key components of all of these systems:

- A mobile application running on a mobile device. This set-up usually requires the implementation of a user- interface that provides concise data in an intuitive and easy-to-use manner, even for non-skilled mobile phone users. Storage capacity and communication with medical sensors and/or remote servers are also required.
- Medical sensors. Remote- or local-monitoring mHealth services require medical sensors to measure vital parameters, such as temperature, blood pressure, and heart rate, among others. Many communication standards are involved here, such as Zigbee, wireless sensors networks, and Bluetooth. For communication with the mobile device, Bluetooth or an appropriate gateway is required. Many mobile devices now have built-in sensors, which may aid in the implementation of remote-monitoring services.
- Wireless communication. To send/receive information from remote servers and/or medical sensors, mHealth systems require wireless communication systems, such as wireless LANs (e.g., WiFi), Bluetooth, or cellular networks (e.g., GSM or 3G). Several mHealth systems require wireless communication to send notifications or alarms to patients. This task is a frequent one in mHealth systems (Fogg, 2009) and can be particularly efficient in developing countries (Danis, 2010).
- Remote servers. Most mHealth systems rely on remote servers to store health information, such as electronic health records and monitored parameters. An appropriate server communication protocol and a database manager are needed.

The work presented in this paper relates to the development of mobile applications required to implement an mHealth system based on the features described above, particularly concerning clinical mHealth applications. Table 1 presents the main features required to implement the tasks needed to develop an mHealth application, according to the Vodafone classification.

Table 1: Features of mHealth systems, according to each category

Category of mHealth system	**Characteristics in common**	**Required mobile platform features**
Education and awareness	-Use communication services, such as SMS or Internet, to send information to users -Present information to user by intuitive interfaces	Remote communication capabilities Support to create user-friendly interfaces
Remote data collection	-Use communication with remote services and databases to store data collected in the mobile device -Use intuitive interfaces to facilitate data collection and present information to users	Remote communication capabilities Support to create user-friendly interfaces
Remote monitoring	-Use communication with medical devices and sensors to gather patient's vital signs -Use communication with remote services to send information and receive feedback -Require persistent storage on the mobile device -Present information to user with graphics or other sophisticated visualization method	Communication with medical devices and/or sensors Remote communication capabilities Persistent storage capabilities Support to create user-friendly interfaces and data visualization features
Communication and training for healthcare workers	-Use communication with remote services to receive training advice or healthcare communication -Present information to user with intuitive interfaces	Remote communication capabilities Support to create user-friendly interfaces
Disease and epidemic outbreak tracking	-Use communication with remote servers or services to send information about diseases or epidemic outbreaks -Use intuitive interfaces to collect data and present information to users	Remote communication capabilities Support to create user-friendly interfaces
Diagnostic and treatment support	-Require communication with medical devices and sensors (some systems) -Store information on the mobile device to retrieve a diagnostic -Use communication with remote servers or services to receive information from healthcare workers -Display information in a friendly interface on the mobile device	Communication with medical devices and/or sensors Persistent storage capabilities Remote communication capabilities Support to create user-friendly interfaces and data visualization features

5. Evaluation Metrics and Case Study

In this section, we describe the evaluation metrics that we considered to be the most important for mHealth applications and services. We then present the application prototypes used for the evaluations.

5.1 Evaluation metrics

Considering the set of features required to implement mHealth applications, as discussed in the previous section, we sought to define the metrics for evaluating the development platforms. These metrics are outlined below:

- Communication with medical devices and sensors. Many mHealth applications and services are intended to be used for remote monitoring and remote data collection purposes. These applications rely on the use of communication technologies with medical devices and sensors, such as wireless sensor networks, Zigbee, and Bluetooth. The last one is a ubiquitous technology present in various systems, including feature and low-budget devices. Bluetooth communication was considered to be an important metric for mHealth applications. We evaluated the availability of this technology in both platforms, relative to the number of APIs supported. To determine the complexity of the implementation, we determined the number of lines of code required for programming. We also evaluated the response times with specific mobile devices.

-Remote server communication. Several categories of mHealth services, including remote monitoring, remote data collection, diagnostic and treatment support, and education and awareness, rely on information stored on a remote server. This situation necessitates the use of remote communication protocols in the mobile device. HTTP is the only mandatory protocol that is supported in all devices, including low-budget devices. We evaluated the availability of APIs to implement HTTP communication and the lines of code required to implement this task. We performed several tests to measure the performance of this type of communication in each platform, using emulators and real devices.

- Persistent storage. All mHealth categories require the mobile device to store information, such as health records, drug schedules, allergy information, or personal information. We evaluated the features included in each platform to provide persistent storage (i.e., available APIs) and the lines of code required to implement these applications.

-User interface. An adequate user interface is an important aspect that will help the user/patient adopt the mHealth services rapidly and use them in a continuous manner.

We evaluated the available APIs to implement user interfaces and explored this feature further through a qualitative evaluation.

5.2 Case study

To evaluate both platforms, we implemented two mHealth application prototypes that fell into the remote-monitoring category, which requires the use of all of the features presented in the previous section. The architecture of the prototypes is depicted in Figure 3. The first prototype was based on an application called the "Cardiac Frequency Monitor" (CFM) (Alaniz 2011), which was previously developed in the Android platform and translated to Java ME. The application provided the same functionalities in Java ME and used the available Java Me APIs to implement the user interface and data visualization. This application was intended to be used by patients suffering from a cardiac disease and requiring continuous monitoring of their cardiac frequency. The patient required the use of a Bluetooth sensor; in this case, we used a Zephyr HxM Bluetooth sensor. The mobile device retrieved information from the sensor every 30 seconds. The signals were stored, and the application reflected the state of the user with different colors (green: ok, yellow: warning, red: danger). Together with a note from the patient, this information was sent to a remote server, to allow data analysis by a healthcare professional. The application also included a module to provide graphical data visualization. The second prototype consisted of a remote-monitoring application to monitor a patient's temperature continuously. We developed this application in both platforms, to test several Bluetooth parameters. To perform this evaluation, we used an analog temperature sensor (LM35) connected to a BlueSentry RN-800-CB interface, which provided Bluetooth communication with the mobile device.

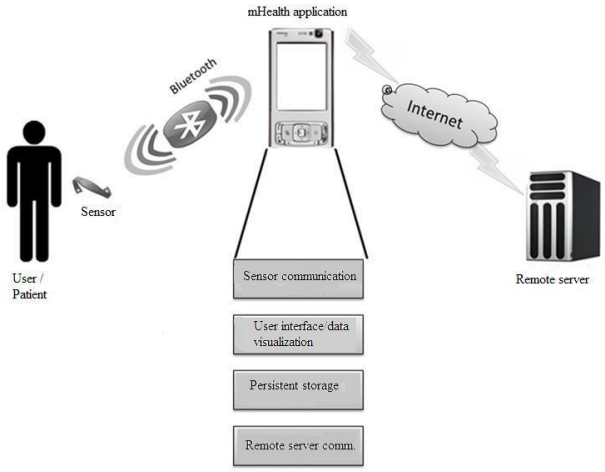


Figure 3. Architecture of the mHealth prototypes

6. Qualitative Evaluation

In this section, we describe the evaluation tests and present the results concerning the metrics presented in the previous section: namely, communication with medical de-vices using Bluetooth, remote server communication using HTTP, persistent storage, and user interface implementation.

6.1 Remote communication, persistent storage and networking

Several tests were performed to evaluate the performance of Bluetooth and the remote communication response times with a server. To evaluate Bluetooth performance, we performed several tests with the real mobile devices, due to lack of support on the Android emulator. We measured the connection time delay with the sensors and the mobile devices, as well as the response times when sending a variable amount of data under several distances. These studies were done with the Zephyr HxM Bluetooth sensor and an analog temperature sensor connected to a BlueSentry device, which provides Bluetooth communication to send several temperature measurements, as depicted in Figure 4.

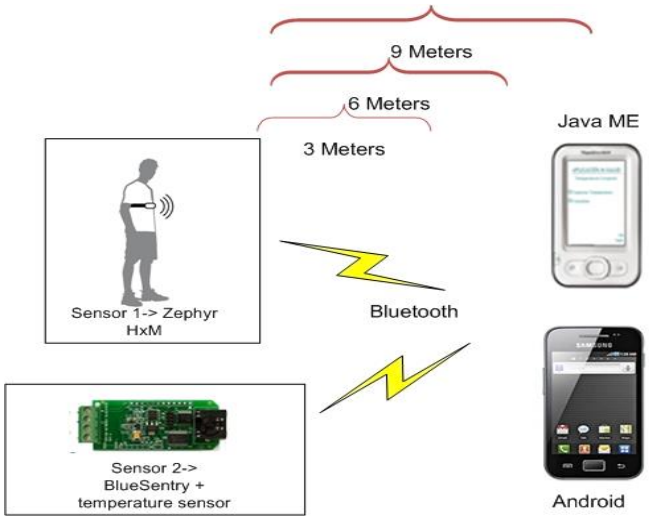


Figure 4. Bluetooth evaluation tests

Figure 5 shows the obtained results. Both platforms had similar performances (difference < 20 ms), with Java ME showing a slightly longer connection time. Although small, this difference could be significant for time-sensitive applications in which real-time measures are mandatory.For remote server communication, we considered the use of HTTP, which is the only mandatory protocol in MIDP devices and is available in all Android devices. We evaluated the connection time to the server, which was the time required to establish communication, including handshake operations. We also sent and received a set of records to the server. These records included measurements taken by the medical sensors. With these operations, we evaluated the writing time to and the lecture time from the server. In all cases, we used an Apache remote server on a Windows PC. We initially performed the tests on the emulators provided by the platforms, which were installed on the same computer.

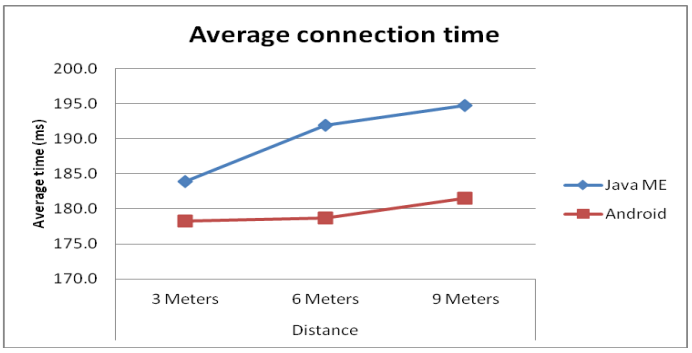


Figure 5. Bluetooth evaluation results

We then repeated the same tests using the real mobile devices. For Java ME, we used the package *javax.microedition.io*. In Android, we used two different options: *apache.org* and *java.net.* As shown in Figure 6, the connection establishment time obtained with *apache.org* in the Android platform was more stable than the times obtained with other packages/platforms. Moreover, this package provided similar results on the emulator and mobile device. In contrast, *java.net* was consistently slower on the mobile device than on the emulator. Java ME performed slower than either of Android's APIs.

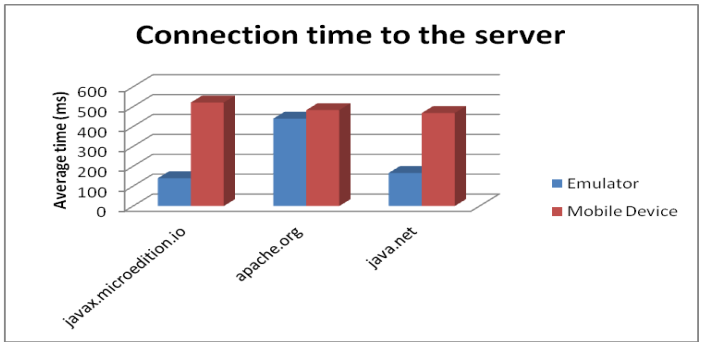


Figure 6. Remote connection time to the server

In terms of writing and reading times to and from the server, Figure 7 shows that *java.net* and *javax.microedition.io* produced similar results, whereas *apache.org* was the slower package. Again Java ME showed marked differences between the emulator and real-device results.

Table 2 shows the analysed features for each platform relative to the features required by mHealth applications. For each feature, we analysed the available APIs and the lines of code required to implement that feature. Both platforms provided APIs to implement all of the required features. The lines of code were similar, because we only analysed the implementation of the basic functionality for each feature.

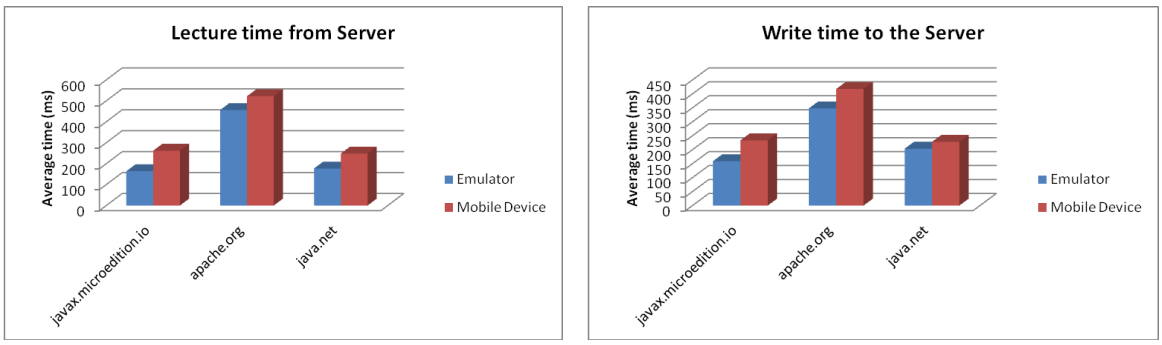


Figure. 7. Average writing/reading time to the server

Some important differences between the platforms were noted:

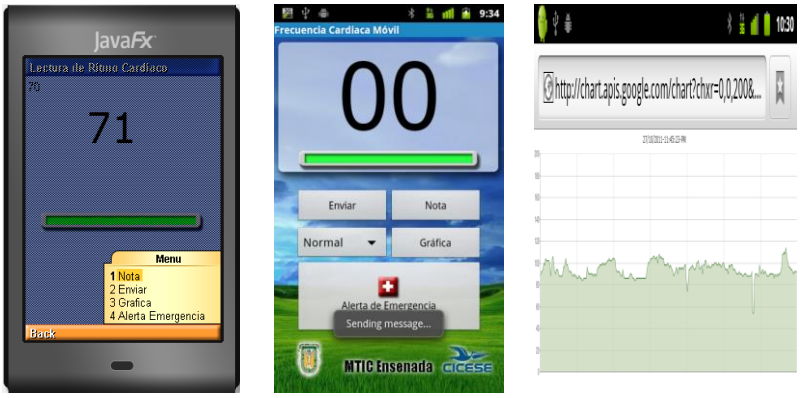
-Bluetooth communication. Android was more limited for this implementation. The Bluetooth standard provides three types of communication: OBEX, L2CAP, and RFCOMM. Java ME provides packages to implement each communication type, whereas Android only supports RFCOMM communication. Besides, there is no Android emulator to perform the tests, being mandatory the use of a real device.
-Remote server communication. Java ME was more limited for this implementation, because it only supports one package. Moreover, remote server communication is not fully implemented on all mobile devices. Android supports its own HTTP API, as well as *apache.org.** and *java.net.*
-Persistent storage. The only mandatory package for MIDP devices is *javax.microedition.rms.* This package supports the storage of a set of records on the mobile device. However, all implementation is left to programmers, no SQL support is offered. Other packages are available, but they are optional (i.e., not all mobile devices support them). On the other hand, Android includes the packages *android.database* and *android.database.sqlite*, which provide a database and allow the use of SQL commands. However, this support requires more memory. Increased memory use can be important in developing countries, where some mobile devices are still very memory-restricted.

Metric	Java ME	Android	Comments
Communication with medical sensors (Bluetooth)			
Available APIs	Bluetooth JSR-82 API includes support for three types of communication: OBEX, L2CAP, RFCOMM. Packages: Javax.bluetooth, Javax.obex	Package: android.bluetooth only supports RFCOMM communication	Lines of code correspond to programming Bluetooth server with RFCOMM and using apache.org.* in Android
Lines of code	9	12	
Remote server communication (http)			
Available APIs	Generic connection framework API Packages: javax.microedition.io	Several packages available, including android.net.http, apache.org.*, and java.net.*	Lines of code correspond to programming Bluetooth server with RFCOMM and using apache.org.* in Android
Lines of code	23	20	
Persistent storage			
Available APIs	No database support Mandatory package: javax.microedition.rms Optional packages: File Connection API JSR-75	Supports SQLite databases android.database.sqlite	Lines of codes required to established RMS/ database connectivity and retrieve one record
Lines of code	21	25	
User interface			
Available APIs	javax.microedition.lcdui (high-level LCDIU components) javax.microedition.ldcui.gamecanvas (low-level programming)	Android.view Android.widget Android.graphics	Android provides the possibility of using Google Charts to display information.
Lines of code	85	JAVA 205 XML 125	

Table 2. Metrics concerning mHealth applications

6.2 User Interface

Figure 8 shows the user interface developed in each platform. Both Java ME and Android rely in the use several forms to navigate through an application. In Java ME, the forms contain several objects, such as TextViews, Buttons, Lists, etc. In Android, the forms are called Activities and consist of a layout containing a set of objects. Unlike Java ME, Android allows the use of XML to construct the user interface, which allows independence of the user interface from the rest of the code that was written in Java. Android also facilitates the use of Google Charts to display data. This ability is useful for a variety of medical information types.



a) b)

Figure 8. Screenshots related to FCM application developed in: a) Java ME and b) Android

Table 3 shows the characteristics of the CFM application in both platforms. In Java ME, the application was smaller compared to the one generated in Android. Although Android offers the possibility of programming the user interface entirely in Java, without using XML files, we chose using XML since it is the recommended option in the Android Reference.

Table 3. Metrics concerning user interface

	Application size (kB)	No. files in application	Lines of code
Java ME	239	5 files (.java)	Java- 429
Android	820	6 files (.java) 9 files (.xml)	Java- 1035 XML- 125

7. Qualitative Evaluation

A qualitative study was conducted in Mexico to evaluate the usability and acceptance of both platforms and to determine whether there was a marked preference for either platform among users. There were 26 participants (13 men) in this user study, aged between 25 to 60 years. 42% of participants had a higher education degree and the rest of them a high school diploma. All participants had previous experience using different mobile applications. Each owned a mobile phone: 30% used Symbian, 15% Android, 15% Blackberry, 10% Windows Mobile, 10% iOS, and 20% did not know what mobile OS they were using. Four participants had a chronic disease, but none of the participants had previously used a health-related mobile application.

To evaluate the platforms, participants were asked to use the CFM application along with the Sephyr HxM Bluetooth sensor. They used the application with the same mobile devices that we used to measure the application performance. Each participant answered a questionnaire, which was developed based on the SUS and CSUQ. The questionnaire consisted of 12 questions, each with 5 possible answers: strongly agree, slightly agree, neutral, slightly disagree, or strongly disagree.
Figure 9 presents the obtained results. Most participants agreed that both applications were easy to learn and answered that the applications were "slightly easy to use" in both platforms. The clearest differences between platforms were obtained for questions regarding whether the platform was "easy to understand" and "well-organized". In both cases, the Android platform performed better than Java ME. When users compared the interfaces and gave their opinion as to which platform they would prefer to use, 77% of users preferred Android and only 23% preferred Java ME. We believe that these preferences were related to the sophisticated user interface tools offered by Android and the possibility of using Google Charts to display information.

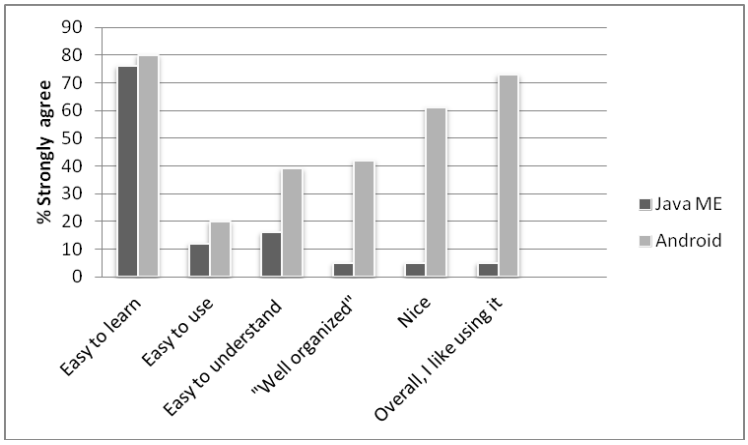


Figure 9. Results obtained from the qualitative evaluation

8. Discussion and Recommendations

Our results show that both platforms provided the features required to implement an mHealth system in developing countries, according to the evaluation metrics considered. Some differences were observed between the platforms:

- In terms of the HTTP communication used to send information to remote servers, Java ME produced similar response times to Android when using *java.net*. When developing such applications, Java ME was more unstable on the emulator than Android, but Java ME was also easier to implement and required fewer lines of code than Android.
-Java ME provided more packages for Bluetooth communication; however, only RFCOMM is mandatory in all devices. OBEX is an optional API. In Android, only RFCOMM is supported. Bluetooth performance was evaluated with the mobile devices, because there was no emulation support in Android. The results indicated better performance on Android; however, because different devices were used, this result was not conclusive.
-Android provides a set of interfaces and packages to construct more complex user interfaces and to display complex information, such as graphics for vital signs, making it more suitable for remote-monitoring or diagnostic systems.
-Android provides more capabilities for persistent storage, because it provides a complete database system using SQLlite.
-Java ME applications had a smaller footprint than Android. Therefore, they were more suitable for very restricted devices.

Each of these features makes one or the other platform better suited for certain mHealth categories, as shown in table 4.

Category of mHealth system	Recommended platform	Comments
Education and awareness	Java ME	Although Android provides the required features to implement this kind of mHealth system, Java ME offers a wider range of mobile devices, allowing the possibility to reach more users.
Remote data collection	Java ME	To reach more users, including users of very-low-budget devices, Java ME is better suited for this kind of mHealth system.
Remote monitoring	Android	Java ME presents some limitations for storing data persistently on the mobile device and for presenting complex information. Android is better suited for these tasks.
Communication and training for healthcare workers	Java Me Android	Both platforms can be used for this kind of system.
Disease and epidemic outbreak tracking	Java Me Android	Both platforms provide the required features to implement this kind of system.
Diagnostic and treatment support	Android	Java ME presents some limitations for storing data persistently on the mobile device and for presenting complex information. Android is better suited for these tasks.

Table 4. Recommended platform for each mHealth category

9. Conclusion

In this paper, we evaluated two Java-based mobile platforms, namely, Java ME and Android, highlighting the platform features and characteristics that are especially useful for the implementation of mHealth services in developing countries. In order to analyze both platforms we characterized mHealth services according to their features related to the categories described by (Vital Wave Consulting, 2009). Furthermore, we implemented and tested a remote-monitoring mHealth system in both platforms, using real mobile devices and medical sensors. Our findings show that even though both platforms can be used to implement such systems, each offers certain functionalities that are more suited to certain categories

Acknowledgements

Thanks to CONACYT for supporting this work by grant 151614 from SEP-CONACYT [SEP (Ministry of Public Education)-Consejo Nacional de Ciencia y Tecnología].

References

[1] Alaníz Sida, J. (2011). Aplicacion Móvil para el Monitoreo de la Frecuencia Cardiaca utilizando el Sistema Operativo ANDROID. Master Thesis, Ensenada B.C.: Universidad Autónoma de Baja California.

[2]Benlamri, R.; Docksteader, L.: MORF: A Mobile Health-Monitoring Platform, IT Professional , Vol.12, No.3, pp.18-25, May-June 2010

[3] Canalys, Press-Release Smart phones overtake client PCs in 2011, 2011, Available on line at: http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011

[4] Danis, Catalina M., Ellis, Jason B., Kellogg, Wendy A., Beijma, Hajo van, Hoefman, Bas, Daniels, Steven D. and Loggers Jan-Willem: Mobile phones for health education in the developing world: SMS as a user interface, *In* Proceedings of the First ACM Symposium on Computing for Development (ACM DEV '10). ACM, New York, NY, USA,, Article 13 , 9 pages, 2010.

[5] Feder, J. Lester. 2010. ―Cell-Phone Medicine Brings Care To Patients In Developing Nations.‖ Health Affairs. 29 (2): 259–263.

[6] Fogg, BJ and Enrique Allen. "10 Uses of texting to improve health". In *Proceedings of the 4th International Conference on Persuasive Technology* (Persuasive '09). ACM, New York, NY, USA, , Article 38 , 6 pages, 2009

[7] Gavalas, D.; Economou, D.,:Development Platforms for Mobile Applications: Status and Trends, Software, IEEE , vol.28, no.1, pp.77-86, Jan.-Feb. 2011, doi: 10.1109/MS.2010.155

[8] Grønli, Tor-Morten, Hansen, Jarle and Ghinea, Gheorghita: Android vs Windows Mobile vs Java ME: a comparative study of mobile development environments. In Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA '10), Fillia Makedon, Ilias Maglogiannis, and Sarantos Kapidakis (Eds.). ACM, New York, NY, USA, Article 45 8 pages. DOI=10.1145/1839294.1839348 http://doi.acm.org/10.1145/1839294.1839348

[9]Helal, S.:Pervasive Java, IEEE Pervasive Computing, 1(1),2002, 82-85.

[10] Istepanian, R.S.H., Zhang, Y.T. : Guest Editorial Introduction to the Special Section on M-Health: Beyond Seamless Mobility and Global Wireless Healthcare Connectivity, IEEE Transactions on Information Technology in BioMedicine, Vol. 8, No. 4, 2004.

[11] Loughborough University "Loughborough University takes mobile phone health monitoring to India", January, 2007, Available on line at: http://www.lboro.ac.uk/service/publicity/news-releases/2007/09_health_monitor.html

[12] Marshall, A., Medvedev, O., and Markarian, G.,: Self management of chronic disease using mobile devices and Bluetooth monitors. In Proceedings of the ICST 2nd international conference on Body area networks (BodyNets '07). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, , Article 22 , 4 pages., 2007
.

[13] Masek, M.; Chang Su Lee; Chiou Peng Lam; Tan, K.T.; Fyneman, A.: Remote home-based ante and post natal care, in Proceedings of 11th International Conference on e-Health Networking, Applications and Services, 2009. Healthcom 2009, pp.60-65, 16-18 Dec. 2009

[14] Meier, R. (2009). Professional Android Application Development. Wiley Publishing, Inc

[15] Mirza F, Norris T., Stockdale R., Mobile technologies and the holistic management of chronic diseases .Health Informatics J. 2008 Dec;14(4):309-21.

[16] M. J. Moron, A. Gomez-Jaime, J. R. Luque, and E. Casilari.: Development and evaluation of a python telecare system based on a bluetooth body area network, EURASIP J. Wirel. Commun. Netw. 2011, Article 2, 10 pages , January 2011.

[17] Mougiakakou, S.G.; Kouris, I.; Iliopoulou, D.; Vazeou, A.; Koutsouris, D.; Mobile technology to empower people with Diabetes Mellitus: Design and development of a mobile application, In Proc. of 9th International Conference on Information Technology and Applications in Biomedicine, 2009. ITAB 2009., pp.1-4, 4-7 Nov. 2009

[18] Nokia, Nokia Data Gathering, March, 2010, Available on line at: https://projects.developer.nokia.com/ndg
[19] Oliver, E. 2009. A survey of platforms for mobile networks research. SIGMOBILE Mob. Comput. Commun. Rev. 12, 4 (February 2009), 56-63. DOI=10.1145/1508285.1508292 http://doi.acm.org/10.1145/1508285.1508292

Author Biography

Mabel Vazquez-Briseno
[mabel.vazquez@uabc.edu.mx]
Received her Ph.D in Computer Science from Telecom SudParis (ex INT) and Pierre et Marie Curie University, France in 2008. She received the M.Sc degree in Electronics and Telecommunications from CICESE Research Center, Mexico, in 2001. She is now researcher-professor at the Autonomous University of Baja California (UABC), where she is a member of the Telematics research group. Her research interests include computer networks, mobile computing and protocols.

Mariana Mendez [mariana.mendez@uabc.edu.mx] received her Master degree in Computer Science from Autonomous University of Baja California (UABC) in 2012. She is now working at the Technology and Engineering Center at UABC in Tijuana, Mexico. Her current research interests include the development of applications and services for mobile devices.

Juan-Ivan Nieto-Hipolito [jnieto@uabc.edu.mx] got his Ph.D. in Computer Architecture and Technology at Technical University of Catalonia, Spain in 2005. He is now researcher-professor at the Autonomous University of Baja California campus Ensenada where he is Coordinator of Post Graduate Studies and leads the Telematics research group.

Elitania Jimenez-Garcia [ejimenez@uabc.edu.mx] received her master degree in Computer Sciences from CICESE research center, Mexico in 2001. She is now professor at the Autonomous University of Baja California, Ensenada, Mexico where she is a member of the Telematics research group

