Technological choices for mobile clinical applications

Ehrler, Frédéric; Issom, David; Lovis, Christian

How to cite

EHRLER, Frédéric, ISSOM, David, LOVIS, Christian. Technological choices for mobile clinical applications. In: Studies in health technology and informatics, 2011, vol. 169, p. 83–87. doi: 10.3233/978-1-60750-806-9-83

This publication URL: https://archive-ouverte.unige.ch/unige:21589

Publication DOI: 10.3233/978-1-60750-806-9-83

User Centred Networked Health Care
A. Moen et al. (Eds.)
IOS Press, 2011

doi:10.3233/978-1-60750-806-9-83

Technological Choices for Mobile Clinical Applications

Frederic EHRLER [a,1], David ISSOM[a], Christian LOVIS [a]
[a] *University Hospitals of Geneva, Division of Medical Information Sciences*

Abstract. The rise of cheaper and more powerful mobile devices make them a new and attractive platform for clinical applications. The interaction paradigm and portability of the device facilitates bedside human-machine interactions. The better accessibility to information and decision-support anywhere in the hospital improves the efficiency and the safety of care processes. In this study, we attempt to find out what are the most appropriate Operating System (OS) and Software Development Kit (SDK) to support the development of clinical applications on mobile devices. The Android platform is a Linux-based, open source platform that has many advantages. Two main SDKs are available on this platform: the native Android and the Adobe Flex SDK. Both of them have interesting features, but the latter has been preferred due its portability at comparable performance and ease of development.

Keywords: EPR, Android, Mobile Health

1. Introduction

Providing care providers real-time, mobile and easy collaborative interactions with the hospital's information system is an important challenge. It is a critical element to improve the efficiency and the safety of care processes [1]. Until recently, these interactions have been limited by devices and interaction models [2]. The new mobile devices represent an important step towards a solution. The development of clinical applications on these devices is not a usual problem of moving an application to a new operating system because of two elements: the pervasive presence of these devices and the disruptive new interaction paradigm introduced by multi-touch screens.

Providing mobile services to physicians requires wise technological choices regarding the platform and the development environment [3]. In the following sections, we first introduce the context in which we started our development research. Then, we present the selection criteria employed to evaluate the candidate technologies. After that, we describe the application we developed to assess the functionality of the candidate SDKs. Finally, we present the advantages and drawbacks of the OSs and SDKs, which we assessed for our development, and what technology we chose to adopt at the end.

[1] Corresponding Author: Frederic Ehrler, University Hospitals of Geneva, Division of Medical Information Sciences, Rue Gabrielle-Perret-Gentil 4, CH-1211 Geneva 14, Switzerland; Email: frederic.ehrler@hcuge.ch.

1.1. Background

The Geneva University Hospitals (HUG) is a consortium federating the public hospitals in the Canton of Geneva, Switzerland. It provides primary, secondary, tertiary and outpatient care for the whole region with 45,000 inpatients and 850,000 outpatient visits a year [4]. The Clinical Information System (CIS) of the HUG is mostly an in-house developed system. It is a service oriented and component-based architecture with a message-based middleware. It is written in Java with J2EE and open frameworks. All exchanges are in SOAP or HTTP/XML [5] [6]. All components building blocks of the CIS, including the ones discussed in this paper are built in such a way that they comply as much as possible with standards, such as IHE (Integrating the Healthcare Enterprise) profiles, so that they are not dependent of any local legacy system. This includes technical, semantically and human-machine interfaces, such as using a terminology server for the language of the interfaces.

2. Method

In order to define the most appropriate technology to develop mobile clinical applications, we defined several criteria organized in three axes:

- **Hardware:** market trends, cost, performance and user acceptance of the mobile devices. Strength of the mobile platform with regards to security, reliability, and privacy.
- **Human:** availability of competent developers on the labor market and existence of a developer community.
- **Software:** complexity of the development environment, cost, user friendliness and reusability of existing and new developments.

It is important to take into account the price of the physical devices supporting the OS. Indeed, when each care provider of the hospital is equipped with a mobile device, a small difference on price becomes really significant. The performance, including power autonomy of the device, is obviously central. Indeed, the good course of the healing process often relies on the real-time access to the relevant information. The information must obviously remain secured as it concerns the private life of the patient.

In addition, we have to consider how quickly developers can master the environment and how easily the work already done inside the CIS can be adapted to the new tools. The choice of widely used languages, such as *ActionScript* or Java, would definitely facilitate the adoption and development as numerous developers are already familiar with these languages. The existence of a professional development environment, the existence of open source projects in this field, and a sufficient developer community, which has already addressed the most obvious questions, also facilitate the developments.

In order to evaluate the features and the ease of development with the different SDKs, we defined a prototype mobile application, sort of test use case, aiming to simplify the care process. With the help of this application, health professionals simply enter the information concerning the patient during the visit instead of recording all the information on laptops. The application is composed of a succession of screens where the user selects the unit, the room, and finally the patient being currently examined. On the last screen, the care provider can enter the vital signs of the selected patient.

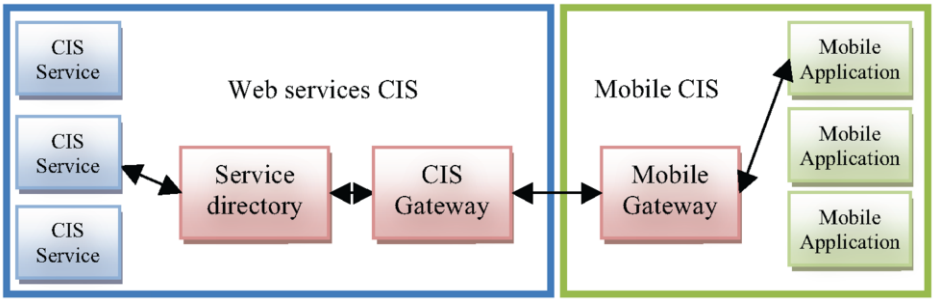

Figure 1. Communication between mobile applications and existing CIS

2.1. Communication Architecture

Regarding the architecture, it was mandatory to think a model that would not create a dependency with any legacy system. Thus, we defined a gateway server providing a centralized access for the mobile application to any required information to or from the CIS. Thus, integrating any mobile application would only require integrating this bridge. It also clearly separates the services that are available remotely from the ones proposed as usual Web services. The gateway server is responsible for formatting the data properly before sending it to the appropriate application on the device. Once the mobile device receives the data, its embedded software is responsible to display the data through its interface and allows the interaction with the user.

Figure 1 shows the link between our mobile application and the current CIS. The services of the existing CIS are externalized through a component named CIS gateway. When a mobile application requires data from the CIS, it communicates with the mobile gateway that transmits the request to the CIS gateway. The service directory is then queried to identify the appropriate service where to retrieve the required information. The information then returns through the same channel. All data transiting through the channel is formatted in XML.

3. Results

3.1. Choice of the OS

The choice of the OS is challenging. There are numerous OS for mobile devices on the market, some of them with marginal shares. In order to simplify the work, it was decided to address only the four that are currently seen as major player, as per the Table 1 next page.

The Apple iPhone is an interesting product as it is widely spread among users [7]. Unfortunately, the development policy of Apple is very restrictive. In addition, the development environment is unique to the OS, thus requiring very specific and devoted skills and education for the development team. Finally, there is a very limited choice of devices, as only the devices provided by Apple are available on the market.

Table 1. Comparison of the principal existing OSs to develop on mobile devices (Market shares of Western Europe, November 2010)

OS	iOS	Symbian	Android	RIM
Developer	Apple	Nokia	Google	Blackberry
Language	Objective-C	C++	Java+XML	Java
Market shares	46.4%	21.77%	15.65%	10.16%

Choosing between Android, Symbian and RIM was trickier. They all possess a significant share on the market, rely on well established language and possess efficient development environment. However, only Android offers all together a huge choice of devices, ranging from very small Smartphone to large tablets, a widespread development environment, a large open source community, and a very transparent development policy.

3.2. Choice of the SDK

One would think it is straightforward to adopt the Android SDK to develop on the Android platform. However, it is worth taking into consideration Adobe, a major actor of the IT world that offers development tools for mobile devices running Android. Adobe provides a SDK named Adobe Flex that has the valuable advantage to generate programs that can be supported by several platforms without any change. We made a quick survey (Table 2) of Adobe Flex and Android SDK characteristics to clarify their benefits and limitations. Some restrictions related to the Flex Hero SDK have been identified. As this SDK is an additional layer over the native SDK, there can be a loss of functionalities. Fortunately, the Flex SDK can handle the main functions required to interact with the mobile device, such as positioning, multi-touch, inclination, etc... The only identified limitation is the impossibility to create Android widgets, but this is not required for our application purpose. The additional layer of the Flex SDK can also induce a reduction of performance. However, we did not observe in our tests and did not found objective and serious studies confirming or infirming this fact.

Regarding the Integrated Development Environment (IDE), the two languages possess a dedicated tool that helps developers generate accurate code. For Android SDK, the Eclipse IDE is perfectly adapted as the code is standard Java language. With the addition of a plug-in, the Eclipse IDE can manage the installed SDK, the documentation, and some drivers to connect the mobile device to the computer. The plug-in offers automated compilation as well an emulator. It allows testing the application locally instead of loading it into the mobile device.

For the Flex SDK, a new version of their development environment, Flex Builder, has been released recently by Adobe to program mobile applications. This IDE based on Eclipse offers programming facility to code in ActionScript and MXML. Like with the Android SDK, there is an emulator that facilitates the development significantly.

Table 2. Comparison of principal existing SDKs to develop on Android platform

Features	Flex Hero SDK	Android SDK
Version	Flex 4.5 Hero	Froyo 2.2
IDE	Flex builder Burrito	Eclipse
Language	ActionScript 3 + MXML	Java+XML
Execution platform	Adobe Compatible	Android

3.3. Comparing Platforms

In order to improve our comparison, we developed our sample application on the two platforms. On the **Figure 2**, it can be seen that there are no strong differences in the human-machine interaction experience between the two interfaces. Both can display and manipulate lists, radio buttons and text inputs and other graphical component.

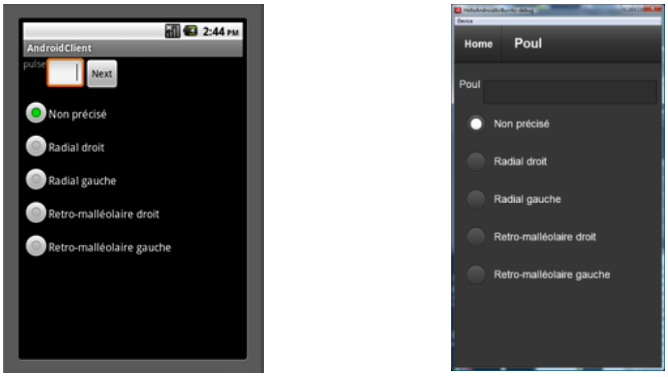

Figure 2. Android SDK and Adobe Flex screens to enter vital signs of the patient.

4. Conclusion

Our constraints, needs and projects, led us to prefer the android OS due its compatibility with the largest number of devices and its open source policy. The selection of the SDK was more difficult as both the Android SDK and the Flex SDK met most needs in terms of features for the development of a mobile application on Android OS. The Flex SDK was finally chosen based on its portability to other platforms at comparable performance and ease of development.

References

[1] Prgomet M, Georgiou A, Westbrook JI. The Impact of Mobile Handheld Technology on Hospital Physicians' Work Practices and Patient Care: A Systematic Review. *JAMIA* **16**(2009), 792-801.
[2] Kubben P. Neurosurgical apps for iPhone, iPod Touch, iPad and Android. *Surg Neurol Int* **22**(2010), 89.
[3] Fischer S, et al. Handheld Computing in Medicine. *JAMIA* (2003), 139–149.
[4] Tschopp M, et al. Computer-based physician order entry: implementation of clinical pathways. *Studies in health technology and informatics* (2009), 673-7.
[5] Borst F, et al. Happy birthday DIOGENE: a hospital information system born 20 years ago. *International Journal of Medical Informatics* **54**(1999), 157-167.
[6] Geissbuhler A, et al. Experience with an XML/HTTP-based federative approach to develop a hospital-wide clinical information system. *Stud Health Technol Inform* **84**(2001), 735-9.
[7] Payne D, Godlee F. The BMJ is on the iPad. *BMJ* **19**(2011).