Cross-platform Mobile Development based on MDA Approach

https://doi.org/10.3991/ijim.v10i4.5570

S. Charkaoui, E. Ben Lahmar, A. Marzak and I. Abdelbaki
Faculty of Sciences Ben M'SIK, Casablanca, Morocco

***Abstract*—Nowadays, the mobile operating systems market (Android, IOS, Windows Phone …) continues to grow. It has become a real challenge for application vendors to provide versatile applications in this competitive market in a short time. The challenge is even greater if the application is planned for multiple platforms whose the operating systems uses different technologies, namely, IOS/Objective-C environment for the Phone and the IPad, Java SDK for Android, etc. This fragmentation makes mobile application development rather difficult and very expensive, hence the use of cross-platform development. To address this later different approach exist, the choice was focused on the MDA approach whose principle is the elaboration of various UML models.**

By studying in detail various target platforms on the basis of a set of criteria and performing a model for each criterion, the aim of our research work is to elaborate a meta-model from UML models realized for each platform.

***Index Terms*—Cross-platform development, Android, Windows Phone 7, MDA approach, IOS.**

I. INTRODUCTION

Currently, in the world of new technology of information and communication, Smartphone and tablets become omnipresent. Otherwise, we cannot evoke Smartphones without talking about mobile applications. Propose mobile applications becomes a strategic issue for companies, result, the mobile application market just keeps getting bigger.

According to Gartner Research Group [1], Android is the first mobile operating system followed by iOS and Windows Phone with an 82.8% market share for android and 14.6% for IOS. As to Windows phone, he resumed his place in the mobile world with a 2.5% market share.

The diversity that exists in the mobile area, including the large number of operating systems that use different technologies, produces a "fragmentation". Recognizing the importance of defragmentation and wanting to optimize the design process of mobile applications, the idea of developing a single application that works everywhere (or almost everywhere) became a goal that was much more difficult to achieve - but remains as attractive as ever. This fragmentation makes mobile application development rather difficult, hence the use of cross-platform development framework.

Several approaches exist to address the cross platform development. In the previous paper an overview is given on the various multiplatform development approaches that exist in the mobile market [2], namely, Source Code Translators, Runtime, JavaScript frameworks, Web-to-native wrapper, App Factory and MDA approach. Only Web-to-native wrapper and MDA approaches allow realizing hybrid applications [3]. We made a detailed study of these approaches in a previous article [2].

The choice of which approach to use depends on three factors, our programming habits, the importance of having an application that appears native and or OS that we touch (IOS, Android ...). Aside from these factors, choosing a model-based approach is another factor that comes to be added to previous, as quoted by Parnas [4], abstraction has always been a key factor to successful software engineering [5].

The Web-to-native wrapper and MDA approaches respond well to the first two factors except that the MDA approach capitalizes on the functional of the application regardless of the technical concerns to facilitate migration [2]. So, much of the application code can be replaced by a model [6].

Our research work is based on the MDA approach [2] to achieve a cross-platform development framework targeting the leading platforms of the mobile market, namely, Android, IOS and windows phone. These mobile platforms don't work in the same way, each one has its own language, APIs, IDEs, etc. This fragmentation makes the development of mobile applications quite difficult and very expensive and This can be observed at all levels, Data Storage, software Architecture, user interface, access to phone data, Communication between applications, … etc.

This article is organized as follows. Section II introduces the three platforms leading mobile market. Section III is devoted to the analysis and modeling of the basic features of mobile platforms. Section IV is devoted to the application of MDA approach for multi-platform development. Section V evaluates our approach. Finally, Section VI concludes the paper and proposes further work.

II. MOBILE OPERATING SYSTEMS

This article examines the three platforms identified as leaders in the mobile market according to Gartner Research Group, Android, IOS and Windows Phone [1]. This section introduces these platforms in terms of structure and system architecture.

A. Structure

Application components are the essential building blocks of an Android application. Among these components, we found The main Activity class that lunches other activities, the main view class, other activity and utility classes, Resources like drawable (bitmaps, icons), layout (CUI layout/content), menu (menu content), values

(string, arrays) and xml (Preferences screen layout & content …) and The Auto-Generated Java Files (R.java File) that contains IDs for all resources. This file is generated to link your resource files for use in your java files in "src" folder.

Components are coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact, specifies the required permissions to use the application and the external APIs used by this later (eg Google Maps Library).

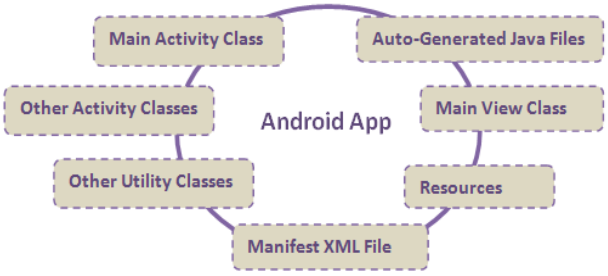

Figure 1. Android App Components

The structure of an Android application is based on many components of different types, which are:

• Activity • Service • Content Provider • Broadcast Receiver. Applications interact via these components and its does not necessarily consists of all four of these components, but to present a graphical user interface there has to be at least an Activity.

Android and WP7 are similar on several points. For both platforms, we find a manifest.xml file, XML views with the code behind, Controls with their events, pages with their life cycle as well as resources of all types.

As Android and windows Phone 7, IOS also has a .plist (property list) configuration file where we define the name of the application, the path to resources and the first page that the user looks at when the application is launched. Apart from this configuration file, an IOS project is also composed of a main.m file which is the application launcher, views represented by .xib files, source code (Controller) in .h (Interface) and .m (Class) files, and a set of libraries provided with the SDK.

B. System architecture

1) Android

The software architecture of Android is based on Runtime, Linux kernel and a set of libraries accessible through an applicative framework [9]. Applicative Framework provides services in the form of java classes for the runtime engine and applications and their management.

Android applications are written in Java, but they are executed by a specific virtual machine called Dalvik. Dalvik and Android execution engine relies on the Linux kernel that handles the interaction with the hardware (drivers and memory management), while a set of APIs provides access to all the services, functions and equipment [10].

2) Windows Phone 7

Windows Phone 7 is an operating system developed by Microsoft and based on Windows CE kernel that takes care for all that is management of system services (security, process management and memory, hardware drivers, and the network stack), the Windows CE kernel takes care.

Windows Phone 7 consists of two major Framework, Silverlight and XNA. Silverlight allows developers to create sophisticated user interfaces whereas XNA is dedicated to creating 3D and 2D games.

Windows Phone applications are written in C # and each application contains a single frame and Pages where the content of the application will be returned.

As android Core Libraries, WP7 provides a base class library (BCL) which includes classes for handling text strings, the Inputs/Outputs management, network communications, Data Access, Threading, IHM …etc.

III. ANALYSIS AND MODELING OF THE BASIC FEATURES OF MOBILE PLATFORMS

Technically, mobile platforms don't work in the same way and don't use the same development languages, user interface, API's, frameworks etc. This fragmentation makes mobile application development rather difficult and very expensive and this can be observed at all levels:

- Data Storage
- Access to phone data (contacts, calendar, etc.)
- Communication between applications
- Access to Phone Features (Phone call, Sending SMS and Emails, etc.)
- Protection and Security (certificate management, Permission …)
- User Interface (GUI)

Given the large number of features, we chose the basic features making difficult the cross-platform mobile development namely, data storage, access to phone data, Communication between applications and access to phone features. In this section we analyze these features and we present the models specific to each feature.

A. Data Storage

Applications can store and retrieve their data in various ways. The solution you choose depends on your specific needs.

1) Android

Android offers several types of storage, exploitable according to your specific needs [12]:

- The data should be private to your application or accessible to other applications.
- How much space your data requires.

We distinguish four ways to save persistent application data [Yin, T13]:

- SharedPreferences: mechanism of configuration data storage of an application into an xml file, this file is useful for recording the preferences of its application. Using this method, we can only store pair key, value.
- Internal storage: Files saved to the internal storage can be set up to allow others to access the files.
- External storage: Files that are supposed to be shared and/or user visible can be stored to the external storage.
- Database storage (SQLite): Android provides full support for SQLite databases. Any databases you

create will be accessible by name to any class in the application, but not outside the application.

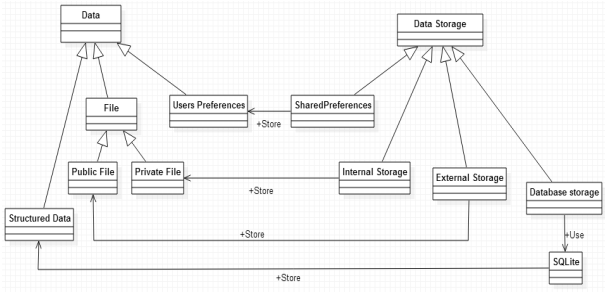

Figure 2. Android's data storage model

2) IOS

IOS offers several data storage policy locally (APIs available), the most important are listed below:

- Property lists: Structured data representation used by Cocoa and Core Foundation as a convenient way to store, organize, and access standard types of data [14].
- SQLite: relational database based on the SQL standard (Structured Query Language) just like MySQL and PostgreSQL. It is an excellent solution for persistent data storage on IOS devices.
- Core Data: is an Object Relational Mapping (ORM) solutions created for IOS to bridge the gap between SQLite and Objective-C. Core data read data from an external source (SQLite database or XML file), by default, the backing store for Core Data is SQLite [15].
- Keychain: is a secure storage to keep user authentication information (IDs and passwords) on the device in a secure and encrypted manner [16].

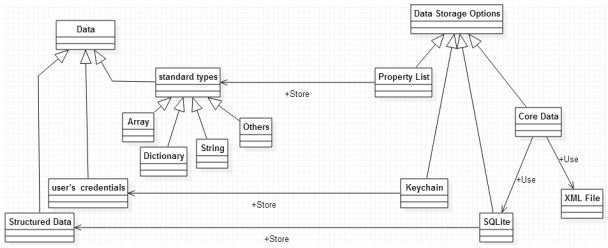

Figure 3. Ios's data storage model

When an IOS application is installed on the IPhone, directories containing everything that the application needs to run are created, below the main directories [17]:

- Document: contains the user documents and sharable application data.
- Preferences : Contains the user preferences based on a properties file mechanism (pfiles)
- Caches: contains data files that may persist between several launches of the application.
- tmp: contains temporary files (which may be removed).
- SQLite: contains necessary libraries for managing a SQLite database.

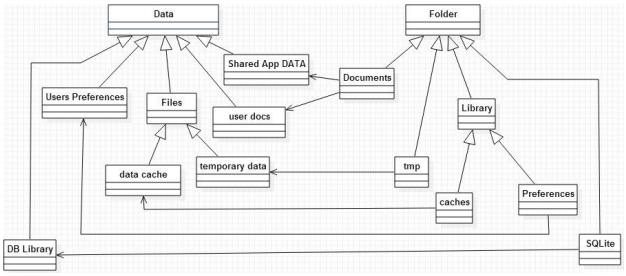

Figure 4. Ios's storage folders model

3) Windows Phone 7

Windows Phone 7 uses isolated storage as visible storage medium only by the application that initialized it, making it impossible by this means sharing information between different applications [18].

As for Android and IOS, Windows Phone 7 also uses a local database integrated in the isolated storage [19].

For the management of data storage, the isolated storage provides two classes:

- IsolatedStorageFile: used for saving data to a file in the app's local folder [20]. It represents the storage space in the isolated storage that contains the files and folders.
- IsolatedStorageSettings : this class is used to save or retrieve data as a key/value pair (Settings) in the isolated storage.

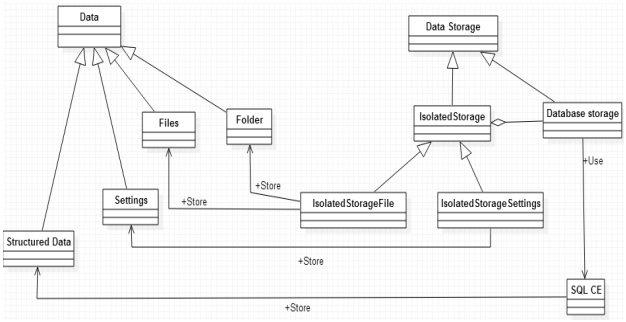

Figure 5. Windows phone's data storage model

To save, display or modify data permanently, without connecting to internet, a mobile system must provide a local database. Android, IOS and Windows Phone 7 perform this task with a relational database (Table 1).

TABLE I.
RELATIONAL DATABASES OF ANDROID, IOS AND WP7

	Android	WP7	IOS
Relational Database	SQLite	SQL CE (Compact Edition)	SQLite

By default, the database of each application is accessible only to the application that created it, but for Android the Content providers provide a mechanism to manage and share this database.

Whereas SQL commands can be executed on Android and IOS since they use the SQLite database. For all operations on the SQL CE database, Windows Phone 7 applications use LINQ (Language Integrated Query) to SQL [19]. We are in a purely object context, no text commands Transact-SQL type.

B. Communication between applications

1) Android

In Android, each application is a process with its own address space. A process can not directly invoke another process but the kernel allows this by using the binder driver.

Android provides a mechanism for interprocess communication (IPC) called binder, for that he modified the linux kernel to add the binder framework that enables the use of the RPC (remote procedure call) mechanism between client and server processes [21]. Remote procedure calls (RPCs) RPC allows client process to call the methods of the server process as if they were local, the result is returned to the client process after the execution (The execution is done in the server process).

The binder framework used by android hides all the complexity of RPC mechanism and provides APIs to simplify the whole interprocess communication mechanism.

Android has chosen to use the binder as IPC mechanism because it has more functionality as sockets and can perform RPC, thing that Pipes does not allow.

To communicate two processes (applications) we must first specify an AIDL interface between client and server processes; this is a file that contains methods shared between the client and the server. These methods represent the entry point of a process to start communication with another. The model of Android's cross Applications communication is presented in the figure 6.

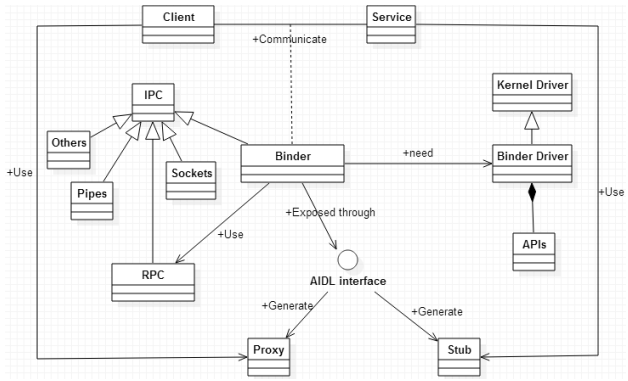

Figure 6. Android's cross Applications communication

2) Windows Phone 7

On the side of Windows Phone 7, communication between applications is extremely limited because of the isolation of applications through the sandbox. The most used ways to communicate is to use web services [22] as a bridge for cross-communication or use the socket IPC mechanism [23].

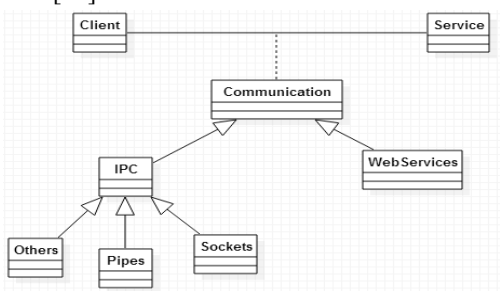

Figure 7. Windows Phone's cross Applications communication

3) IOS

In IOS, communication between applications isn't done in a straightforward manner. The only medium where the communication is performed is when the application communicates with the system which in turn communicates with the other application. IOS provides a form of interprocess communication (IPC) between applications by using the handlers protocol (URL Schemes) [24]. Other IPC mechanisms can be used as, Distributed Objects, XPC, AppleEvents, etc, but remains the most used.

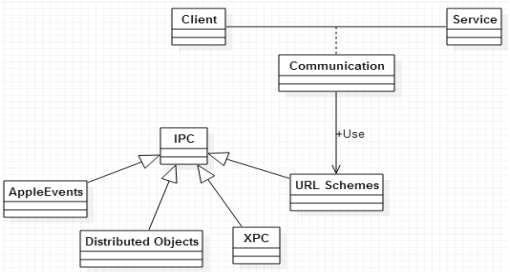

Figure 8. IOS's cross Applications communication

C. Access Phone Data

1) Address book

a) Android

Access to contacts (address book) in Android is done via the Contacts Provider [25], but first of all, we need to add in the manifest file the permission that allows us to access contacts:

android.permission.READ_CONTACTS

android.permission.WRITE_CONTACTS

android.permission.GET_CONTACTS

The Contacts Provider is one of the components of android, he takes care of the management of the central repository of phone data [26]. In an application you can access the information of the Contacts Provider directly by calling the methods of ContentResolver class, this class allows us to [25]:

- retrieving lists of contacts [27]
- displaying the details for a particular contact
- modifying contacts using intents

b) Windows Phone 7

To access the Contacts on Windows Phone 7, we need to add a reference to the Contacts object (Microsoft.Phone.UserData) and make an asynchronous search [28].

Using the ID_CAP_CONTACTS user permission is required for access to the application contact data [29].

c) IOS

IOS is based on the Contacts framework which provides Objective-C and Swift APIs to access the user's contact information. This framework replaces the framework AddressBook [30].

2) Calendar

a) Android

To manipulate calendar data, android uses the Calendar Provider API [31] which is the repository of user's calendar events.

As for access to phone contacts, we need to add the permissions that will allow us to do operations on the calendar:

android.permission.READ_CALENDAR

android.permission.WRITE_CALENDAR

b) Windows Phone 7

To access calendar data on Windows Phone 7, we need to add a reference to the Appointments object (Microsoft.Phone.UserData) and make an asynchronous search [32] without forgetting to add the ID_CAP_APPOINTMENTS user permission [29].

c) IOS

In IOS, the manipulation of the Calendar application data is done via the Event Kit framework [33]. The Calendar application information is stored in the calendar database.

The following table summarizes access to phone data (Calendar, Contacts) for Android, IOS and WP7.

TABLE II.
ACCESS PHONE DATA

	Contacts	Calendar
Android	-Contacts Provider component -Use ContentResolver class -Permissions according to your specific need : android.permission.READ_CONTACTS android.permission.WRITE_CONTACTS android.permission.GET_CONTACTS	-Calendar Provider API - Permissions according to your specific need : android.permission.READ_CALENDAR android.permission.WRITE_CALENDAR
IOS	-Framework Contacts (Provide Objectif-C and Swift APIs)	-Framework Event Kit -Calendar database
Windows Phone 7	Reference to Contacts object Add ID_CAP_CONTACTS permission	Reference to Appointments objects ID_CAP_APPOINTMENTS permission

D. Access phone features

All mobile phones (Windows Phone 7, Android, IOS ...) have different types of sensors. In this article we are interested to the basic sensors, namely, the accelerometer (Motion sensor) and the GPS (Position Sensor). To access sensors available in the phone, each platform has its own way.

1) Accelerometer

Android uses the sensor framework [34] which provides several classes and interfaces that can help to perform various tasks related to sensors (Sensors available in the phone, capabilities of a sensor ….). Ios on his side uses the CoreMotion framework [35] and windows phone 7 uses the reference of Sensors object (Microsoft.Devices.Sensors) [36].

2) GPS

Android provides a mechanism for accessing the sensors and the sensor data using the android.hardware package, except of GPS which is accessible via the Android location services using the Location API [37].

To get the exact geolocation of the device, IOS relies on the Core Location framework [38] and windows phone 7 uses the GeoCoordinateWatcher class of the reference of the Location object [39].

TABLE III.
BASIC SENSORS

	Accelerometer	GPS Location
Android	Utiliser le framework Sensor android.hardware.SensorManager android.hardware.SensorEventListener	android.Location.LocationManager; android.Location.Location
Windows Phone 7	Microsoft.Devices.Sensors.Accelerometer	System.Device.Location.GeoCoordinateWatcher
IOS	Utiliser le framework CoreMotion Classe CMMotionManager	framework CoreLocation Utiliser la classe CLLocationManager

E. Other Features

Aside from the sensors (GPS, accelerometer), access to native features like Phone call, Send SMS, Send Emails, photo capture, etc…, also interest us. Table 4 presents the main features.

Unlike Android and IOS, Windows Phone 7 doesn't allow direct access to the phone native functionality. The only way is to go through Launchers and Choosers.

TABLE IV.
NATIVE FEATURES

	Android	IOS	Windows Phone 7
Phone call	Use Intent.ACTION_CALL startActivity(new Intent(Intent.ACTION_CALL, Uri.parse('tel:523')))	openURL method of the object [UIApplication sharedApplication]	Microsoft.Phone.Task .PhoneCallTask
Send SMS	android.telephony.SmsManager	Importing the MessageUI Framework Use Class : MFMessageComposeViewController	Microsoft.Phone.Task .SmsComposeTask
Photo capture	Use the API android.hardware.camera Use Intent class : new Intent(MediaStore.ACTION_IMAGE_CAPTURE)	UIImagePickerController	Microsoft.Phone.Task .CameraCaptureTask
Send Email	use Intent.ACTION_SEND	Use MFMailComposeViewController class of the MessageUI Framework	Microsoft.Phone.Task .EmailComposeTask

F. Discussion

Android, IOS and Windows Phone are the leader platforms in the mobile world. These three platforms don't work in the same way, they don't use the same development languages and differ on several points, namely data storage, communication between applications, access to phone data … etc.

Table 5 is divided into two parts. The first, presents the main correspondences between Android, IOS and Windows Phone 7 (WP7) whereas the second is dedicated to the various criteria discussed in section III.

Android, IOS and WP7 are similar on several points. In terms of basic structure, the three platforms have a configuration file, XML/XAML/XIB views with code behind, Controls with their events, pages with their life cycle and resources of all types.

To save, display or modify data permanently, without connecting to the internet, the three mobile systems perform this task with a relational database. Android and IOS use the same database, namely SQLite whereas WP7 uses SQL CE. For other data storage means, each platform has

its own storage system, figures 2, 3 and 5. Table 6 summarizes the various storage means.

TABLE V.
ANDROID, IOS AND WP7 CORRESPONDENCES

	Android	IOS	WP7
Main correspondences			
Language	Java	Objective C	C#
Configuration file	Manifest File .XML	.plist (property list) File	Manifest File .XML
Libraries	API	Framework (use in the project)	Reference (Add to project)
UI	UI XML	NIB File	XAML File
Code behind	File.java	.h and .m Files	File.cs
Main Screen	Main Activity	Window (UIWIndows)	Frame
screen	Activity	View (UIView)	Page
Model	MVC, MVVM, MVP ...	MVC	MVVM
Web browser	Webview	UIWebView	WebBrowser
Relational Database	SQLite	SQLite	SQL CE (Compact Edition)
Data base Access	SQL	SQL	LINQ to SQL
Criteria			
Data storage	Yes	Yes	Yes
Cross App Communication	Binder mechanism of IPC (Inter-process communication)	Extremely limited communication. - Services web - IPC	URL Schemes mechanism of IPC
Phone Data	Yes, through APIs (Contacts Provider, Calendar Provider)	Yes, through Frameworks (Contacts, Event Kit)	Yes, Use References to Contacts and Appointments object
Access to native functionality	Yes	Yes	No, indirect access through Launchers and Choosers

TABLE VI.
DATA STORAGE OPTIONS

	Android	WP7	IOS
Settings	SharedPreferences	IsolatedStorageSettings	property lists
Files And Folders	Internal/External Storage	IsolatedStorageFile	
Relational Database	SQLite	SQL CE (Compact Edition)	SQLite

Google's Android is the most widely used platform for Smartphones and tablets in the mobile market, but, in terms of security the researches comparing the degrees of safety between platforms, have concluded that Ios is the most secure platform [40]. Li and Clark explain in their mobile security article that IOS and Android isolates each application from accessing unauthorized data [41]. Some platforms like Android, IOS and WP7 isolate an application of another and phone resources (personal contacts, the user's location, camera access, audio, Internet access …) using the sandbox system. To access phone resources, the sandbox system implements a permissions system that forces users to grant permissions (permissions for Android, capabilities for Windows Phone 7 and prompts for IOS) [42].

Another point where our three philosophies clash is communication between applications. The three platforms use interprocess communication (IPC) for communicating multiple applications. Android offers a new mechanism called binder, it has more functionality as sockets and can perform RPC, thing that the Pipes mechanism doesn't allow. IOS on his side provides a form of interprocess communication (IPC) between applications using the handlers protocol (URL Schemes) [43] while the Windows Phone 7 allows the use of all classic IPC mechanisms (Socket, Pipes …)

Access to phone features is among the criteria considered in this article. Android and IOS allow a direct access to the phone native features (sending SMS, e-mails ...) whereas WP7 doesn't allow direct access but only through Launchers and Choosers.

IV. APPLICATION OF MDA APPROACH FOR MULTI-PLATFORM DEVELOPMENT

After evaluating different approaches to cross-platform mobile development that we have detailed in the Article [2], following a multi-platform development approach based on models is the future of applications. The MDA approach responds well to our need, its key principle consists of the use of models in the various phases of the application development cycle. Three levels of models representing the levels of abstraction of the application, the CIM (models requirement), the PIM (analysis and design) and PSM (code). MDA believes that the analysis and design models must be independent of any implementation platform, whether Java, .Net, Objectif-C, etc.

Elaborate models (PSM) following the analysis of the features mentioned in section III of each platform is the first step of the process of realization of the multi-platform development framework. The main goal is to establish from these models (PSM) a general meta model (PIM) according to business logic independently of the implementation technology, it represents the second level of abstraction defined by the MDA approach (figure 2).

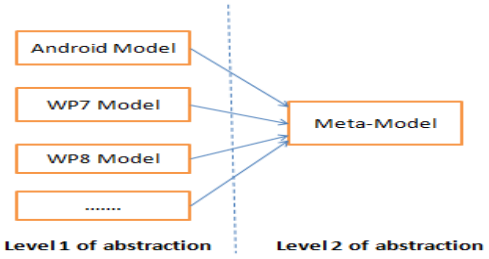

Figure 9. Meta-Model representation

The designed models represent the PSM model of the MDA approach, they are made based on the UML language (Unified Modeling Language), as it is recommended by the MDA approach as the language to use to carry out analysis and design models that are independent of implementation platforms. Indeed, MDA isn't definitively connected with this language. If tomorrow a new language appears to replace UML, it will be always possible to transform UML models to this new language.

The MDA approach will allow us to automatically generate source code from a UML model and this through a model transformation, as shown in the following figure.

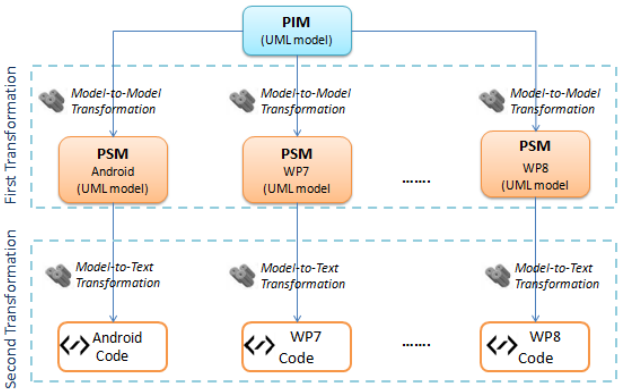

Figure 10. Model transformation

Four types of model-to-model transformations are proposed by the MDA approach:

- PIM-to-PIM transformations: used for the refinement of the analysis and design model.
- PIM-to-PSM transformations: are used to project a PIM to the selected execution infrastructure.
- PSM-to-PSM transformations: relate to platform dependent model refinement.
- PSM-to-PIM transformations: abstract models of existing implementations into platform independent models.

V. VALIDATION

The main goal of this research is to transform PSM models of Section III in a PIM model (abstract representation) by applying the PSM-to-PIM transformations. Knowing that the models are an abstract entity which doesn't require computer representation to exist, MDA using the models for the purpose of productivity, it is however necessary they have concrete representations in order to be handled by computer.

Two different ways are defined by MDA to represent models by computer, in the form of textual documents or programming objects. A representation in the form of programming objects is more suited to computer manipulation (transformation, execution, validation, etc.), while the representation as textual documents is more suited to the storage of templates on hard disk or the exchange of models between applications. This type of representation is defined by the XMI standard (XML Metadata Interchange) which is a way to represent the models in the form of XML documents. The principle of operation of this latter is to automatically generate the structure of representation formats of models from their meta-model.

To validate our approach and taking into consideration our mastery of XMI standard, we chose the representation in the form of textual documents. Applying XMI to UML enables the automatic generation of a DTD that allows represent UML models as XML documents. In other words, to represent the model in the form of XML document, XMI generates its DTD from the meta-model of the model. The XML document that defines the model is structured by the DTD generated.

The figure below illustrates the transformation of a model described in UML to an XML document from an XMI definition.

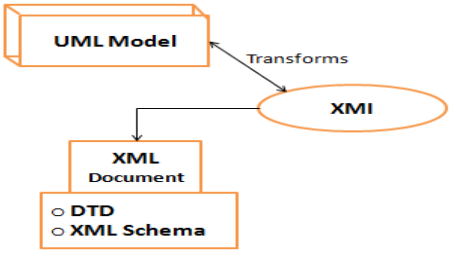

Figure 11. Transformation of a business model

In summary, in order to represent the model as an XML document XMI generates its DTD from the model. The XML document that defines the model is structured by the generated DTD.

After presenting the model in XMI, the next step is to generate the source code through a modeling tool taking as input the XMI representation defined previously. So we got an API with multiple interfaces, each feature is represented by an interface, this latter provides all the methods for manipulating the corresponding functionality.

VI. CONCLUSION

In the context of achieving a cross-platform mobile development framework, an MDA based approach to multi-platform mobile application modeling has been proposed. In this paper, we provide analysis and modeling of functionality of three different platforms, Android, IOS and windows phone 7 to elaborate a framework allowing us to develop applications on different mobile OS's. The functionalities studied are: data storage, access to phone data, communication between applications and access to phone features. For each platform we perform a UML model that will help us later in other research work to elaborate a meta-model of cross-platform mobile development framework.

In future works we will extend our comparison providing more details on each functionality without forgetting to analyze other important functionality which is the software architecture.

REFERENCES

[1] Kishalaya Kundu, « Android & iOS Together Hold 96.8% Of The Market », in androidheadlines.com, 2015.

[2] Charkaoui Salma, Abdelbaki Issam, Ben Lahmar El habib and Marzak Abdelaziz, "Towards a Multi-Platform Development Based on MDA Approach" in nternational Journal of Computer Networks and Communications Security (ijcncs), VOL. 3, NO. 3, pages 103–109, MARCH 2015.

[3] Salma Charkaoui, Zakaria Adraoui, El Habib Benlahmar, « Cross-platform mobile development approaches » in Conference on Information Systems and Technology (CIST), 2014, p188-191. http://dx.doi.org/10.1109/cist.2014.7016616

[4] D. L. Parnas, « On the criteria to be used in decomposing systems into modules » in Commun. ACM, 15(12):1053–1058, 1972. http://dx.doi.org/10.1145/361598.361623

[5] G. Lehmann, M. Blumendorf, F. Trollmann, and S. Albayrak. MetaModeling Runtime Models. In Models@run.time Workshop at MoDELS 2010, volume 6627 of LNCS. Springer, 2010.

[6] Le Goaer.O, Barbier.F, Cariou.E, Pierre.S, «Android Executable Modeling: Beyond Android Programming » in Conference on Future Internet of Things and Cloud (FiCloud), Barcelona 2014, p 411 – 414.

[7] Jean Vanderdonckt, « A MDA-Compliant Environment for Developing User Interfaces of Information Systems », in 17th International Conference Advanced Information Systems Engineering (CAiSE), Porto Portugal 2005, p16-31. http://dx.doi.org/10.1007/11431855_2

[8] Michael Nebeling, Theano Mintsi, Maria Husmann, Moira Norrie, « Interactive development of cross-device user interfaces », in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2014, p2793-2802. http://dx.doi.org/10.1145/2556288.2556980

[9] Richa Sharma, "Development Platforms for Mobile Applications:Status and Trends", in International Journal of Computers & Technology (ijct), pages 52-54, Volume 3. No. 1, AUG, 2012.

[10] Erika Chin, Adrienne Porter Felt, Kate Greenwood, David Wagner, « Analyzing inter-application communication », in Android, in MobiSys '11 Proceedings of the 9th international conference on Mobile systems, applications, and services, 2014, p 239-252.

[11] Aijaz Ahmad Sheikh, Prince Tehseen Ganai, Nisar Ahmad Malik, Khursheed Ahmad Dar, « The SIJ Transactions » in Computer Science Engineering & its Applications (CSEA), Vol. 1, No. 4, September-October 2013.

[12] Android, « Storage Options », in http://developer.android.com.

[13] YIN Jinghua,WANG Huajun, "Data storage based on android development", in Digital Communication, 2013.

[14] Apple, in « About Property Lists », in https://developer.apple.com.

[15] Apple, « What Is Core Data », in https://developer.apple.com.
[16] Apple, « Keychain Services Concepts », in https://developer.apple.com.
[17] Apple, « iOS Data Storage Guidelines », in https://developer.apple.com.
[18] Misrosoft, « Isolated Storage w Windows Phone 7 », in https://msdn.microsoft.com.
[19] Misrosoft, « Local database for Windows Phone 8 », in https://msdn.microsoft.com.
[20] Misrosoft, « Quickstart: Working with files and folders in Windows Phone 7 », in https://msdn.microsoft.com.
[21] Android, « Processes and Threads », in http://developer.android.com.
[22] Misrosoft, « Communications for Windows Phone 8 », in https://msdn.microsoft.com.
[23] Misrosoft, « Local Interprocess Communications », in https://msdn.microsoft.com.
[24] Tielei Wang, Kangjie Lu, Long Lu, Simon Chung, Wenke Lee, « Jekyll on iOS: when benign apps become evil », in SEC'13 Proceedings of the 22nd USENIX conference on Security, p 559-572.
[25] Android « Accessing contacts Data », in http://developer.android.com.
[26] Android « Contacts Provider », in http://developer.android.com.
[27] Android « Retrieving a List of Contacts », in http://developer.android.com.
[28] Misrosoft, « How to access contact data for Windows Phone 8 », in https://msdn.microsoft.com.
[29] Misrosoft, « App capabilities and hardware requirements for Windows Phone 8 », in https://msdn.microsoft.com.
[30] Apple, « Address Book Framework Reference for iOS », in https://developer.apple.com.
[31] Android « Calendar Provider », in http://developer.android.com.
[32] Misrosoft, « How to access calendar data for Windows Phone 8 », in https://msdn.microsoft.com.
[33] Apple, « Introduction to Calendars and Reminders », in https://developer.apple.com.
[34] Android « Sensors Overview », in http://developer.android.com.
[35] Apple, « Core Motion Framework Reference », in https://developer.apple.com.
[36] Misrosoft, « How to get data from the accelerometer sensor for Windows Phone 8 », in https://msdn.microsoft.com.
[37] Android « Location Strategies », in http://developer.android.com.
[38] Apple, « Getting the User's Location », in https://developer.apple.com.
[39] Misrosoft, « GeoCoordinateWatcher, classe », in https://msdn.microsoft.com.
[40] Andersson, Tobias, Johansson, Erik, « A closer look and comparison of cross-platform development environment for smartphones », in Digital Vetenskapliga Arkivet (DiVA), 2013, p8.
[41] Qing Li, Clark.G, «Mobile Security: A Look Ahead », in Security & Privacy, IEEE (Volume: 11 , Issue: 1) 2013, p 78 – 81.
[42] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang,Phillipa Gill,David Lie, « a look at smartphone permission models », in workshop on Security and privacy in smartphones and mobile devices, p63-68.
[43] Luyi Xing, Xiaolong Bai, Tongxin Li, XiaoFeng Wang, Kai Chen, Xiaojing Liao, Shi-Min Hu, Xinhui Han, « Unauthorized Cross-App Resource Access on MAC OS~X and iOS », in Conference on Computer and Communications Security (CCS), 2015, p31-43.

AUTHORS

S. Charkaoui, E. Ben Lahmar, A. Marzak, I. Abdelbaki Authors: Department of mathematics and informatics , Faculty of Sciences Ben M'SIK, Casablanca, Morocco (e-mail: charkaoui.salma@gmail.com, h.benlahmer@gmail.com, marzak@hotmail.com, i.abdelbaki@gmail.com).

Submitted 18 February 2016. Published as resubmitted by the authors 23 September 2016.