Modeling and Code Generation of Android Applications Using Acceleo

Hanane BENOUDA1*, Redouane ESSBAI1, Mostafa AZIZI1 and Mimoun MOUSSAOUI1

1 MATSI Laboratory, EST Oujda, Mohammed First University (UMP), Oujda, Morocco
{benouda.89, Es-redouane}@gmail.com, {azizi.mos, m.moussaoui}@ump.ma

Abstract

The development of mobile applications becomes increasingly popular in our daily life due to the intensive use of applications in mobile devices like smartphones and tablets. The diversity and variety of mobile operating systems (Android, iOS, Black Berry, Windows Phone, etc.) make software engineers in front of a big challenge to develop the same application for these different platforms. This paper presents a methodology based on the Model Driven Architecture (MDA) to develop mobile applications according to the principle "develops once, use everywhere". Our approach exploits UML modeling and Acceleo to generate specific code in order to accelerate and facilitate the development of mobile applications.

Keywords: Software engineering, MDA, UML, Acceleo, code generation, mobile applications

1. Introduction

Smartphones and tablets become an alternative to personal computers with different Operating Systems (Android, iOS, Black Berry, Windows Phone, etc.). Each platform uses different programming languages and tools. The development of the same application for these different mobile technologies is really an exhausting task. This diversity of development languages and tools makes difficult developing cross-platform applications. Thus, it requires software engineers to make a choice on the platform, while ensuring the largest dissemination. The majority of these mobile devices with touch screens run Android OS. Android [6] is the Google’s mobile platform, created in November 2007 and it is open source. Android Applications are developed using Java language, but these don’t run on the traditional JVM (Java Virtual Machine). Also, a desktop Java applications developed for desktop must be adapted principally due to new concepts, as Service and Activity, and specific APIs used in the development of Android applications.

Among these approaches, Model Driven Architecture (MDA) has received attention because it provides abstraction through high level model like UML [4], thus it facilitates the design of complicated and complex software.

The main objective of the model Driven Architecture is to reduce the difference between the problem domain and the software implementation by the use of technologies which support systematic model transformation. MDA provides significant advances to control the development of software application and to allow productivity gains, increased reliability, and important improvement in sustainability and good ways to treat changing constraints. MDA [1] considers models as key artifacts, and these are used in all stages of software engineering which are the analysis, the design, the implementation and the

* Corresponding Author
testing stage. The goal is to switch from mainly documentary models to productive models, using the definition of a number of operations on these models to produce software applications.

The model transformation, the models checking and the reverse engineering are among those operations. Its object is to make automatic transformations of models up to the generation of the code which implements the software. Model Driven Architecture approach expects to replace the slogan “Write once, Run everywhere” by “Model once, Generate anywhere”.

This paper proposes a MDA approach for Android applications development. This approach includes UML modeling and automatic code generation using Acceleo with the aim to accelerate and facilitate the development of Android Applications.

This paper is organized as follows: the second section defines the MDA approach, while the third presents some related works. Android applications are the main topic of the fourth section. In the fifth section, we present the MDA approach for android applications. In the sixth section, we discuss the applicability of our approach through a case study.

2. Model Driven Architecture (MDA)

The Object Management Group (OMG) had made public the MDA in 2001 as an approach for Model Driven Engineering (MDE). In this approach, models have taken more focus in the development process and since it has gained an increasing interest from both industrial and research laboratories. MDA separates the functional specification and the modeling task of a system from the implementation details on a given platform, without losing the integration of the model in a target platform. The MDA considers the models to transform as a productive element used to automatically generate the application source code. The main technologies of MDA approach are Unified Modeling Language (UML) [4], Meta-Object Facility (MOF) [2], XML Meta-Data interchange (XMI), Query/View/Transformation (QVT) [2] and Acceleo [5].

The OMG has defined certain terms around the models named meta-metamodel, metamodel and model, and MDA defines also three different levels of models: Computation Independent Model (CIM), Platform Independent Model (PIM), and Platform Specific Models (PSM) for modeling the application and then successive transformations to generate source code. These models are illustrated in Figure 1 and Figure 2.

![Figure 1. Four-Level Architecture of OMG](image-url)
Thus, the MDA standard produces target models from a number of source models and allows for the same model to be implemented on multiple platforms through standardized projections. It allows applications to interoperate with models and support the development of new techniques and platforms. The implementation of the MDA is completely based on the models and their transformations see Figure 2.

![Figure 2. MDA Models [14]](image)

According to the MDA approach, model transformation provides a mean to specify the way to produce target models from source models. Model transformations are composed of rules and constraints that transform elements defined in a source metamodel into other elements of the target metamodel. Therefore, transformations are applied to source model conforming to the source metamodel and generate a target model conforming to the metamodel. See Figure 3.

![Figure 3. Transformation Process in the MDA standard [14]](image)

Model transformation is the transaction of converting one model to another in the same system. The transformation produces a platform specific model (PSM) using the combination of the platform independent model (PIM) and additional information. Whatever the ultimate target platform can be the first step when constructing MDA-based applications is to create a PIM expressed via UML. This general model can then be transformed into one or more PSM such as .NET, EJB, etc.

An MDA mapping supplies specifications for the way to transform a PIM into a particular PSM. The target platform model defines the nature of the mapping. While part
of the transformation may result from a manual exercise, the intention is clearly to automate a lot of process as allowed by the tools in use.

A model transformation mapping must be defined using some language, be it an action language, a natural language, or a dedicated mapping language. The transformations between the various models are produced using tools compatible with the OMG standard named MOF QVT [2].

A record of the transformation must include a chart to indicate which PIM elements were mapped to which PSM elements and comprise the mapping rules employed for each part of the transformation. The generation of code is the final step for the transform process so the fully the run-time and application semantics may be included in the PSM.

3. Related Works

The MDA approach has demonstrated itself for the development of enterprise applications and may also bring a lot for mobile applications. The diversity of existing mobile platforms, variety of devices and heir multiple features; present an important challenge to develop mobile applications in such environments. The MDA approach helps us ensure the gain productivity and sustainability of expertise while addressing the issues of fragmentation of mobile platform. Recently, several studies have been interested in this direction.

In [11], the authors contributed to improvement of the generation of graphical user interfaces (GUI) for many platforms as Java Server Faces (JSF) and Java Server Pages Standard Tag Library (JSTL) by the use of the AndroMDA open source Framework. Their paper describes the approach based on the analysis and design of the PIM model by using UML diagram, and then enriched with stereotypes to get the model PSM, once the PIM to PSM transformation is realized. Then, the specific code of the target platform is generated by AndroMDA.

[21] presents an UML profile for the modeling of user interfaces. The authors use the MDA approach to define a PIM, then enriched by stereotypes in order to get a PSM by using M2M transformation. The last model is transformed to source code further to Model to Code transformations. In order to do this, the researchers use UML diagrams to define the PIM and the QVT to produce the different transformations.

[14] proposes an approach to design of the user interface (UI) of mobile applications. The researchers use the MDA approach to provide a PIM model under textual format and the M2M (Model to Model) and M2T (model to Text) transformations are applied to generate the GUI for a specific platform. In order to do this, the authors utilize Xtext to define a DSL and Xtend 2 to make different transformations.

[18] presents a comparison of existing cross-platform tools that relieve the development efforts, rejecting the web apps which are too naïve. The authors introduced their language called Xmob dedicated to mobile software engineering, but not experts on each platform, and can be cross-compiled through model transformations, as required by the MDA approach. The solution supporting the architecture of the Xmob cross-compile is started with Xmob and ended to native code, every transformation chain contains two M2M transformations and one M2T transformation:

- PIM to PSM: transformation of high-level Domain Specific Language (DSL) concepts into equivalent.
- PSM to PSM (optional): iterative introduction of good practices (various refactoring, naming conventions, etc.).
- PSM to Code production of source code which corresponds to the elements in the PSM.
In [7], authors use the implementation of the MDA approach in order to model and generate graphical interfaces for mobile platforms. This approach consists of three main steps:

- Modeling of the graphical interface with UML,
- Transformation of diagrams obtained in a simple plan XML by the use of JDOM (Java Document Object Model) API,
- Generation of the graphical interface on the base of the approach MDA.

This method presents the benefit of automatically generating graphical interfaces for various mobile platforms from a UML model, and the MDA transformation rules are expressed in the ATL (Atlas Transformation Language).

In our paper, we present a new approach based the MDA approach to model and generate mobile applications. The MDA approach provides a PIM model and transforms it into a PSM model then generates code source for our mobile application. In order to do this, our method uses the translationist approach and presents the benefit of automatically generating code for a mobile application from the UML model, because UML is a standard language to specify, visualize, construct, and document the artifact of software systems, and it is a set of best engineering practices having been proven in the modeling of complex systems. The MDA transformation rules are expressed in the Query/ View/ Transformation (QVT) [2] and the M2T transformation using Acceleo [5].

4. Android Applications

Android is an OS based on the Linux Kernel designed for mobile devices such as smartphones and tablets. The android applications are developed in Java, but run on a specific virtual machine named Dalvik. This platform supplies also an API with diverse components that can be reused to define an Android application. Between them, activity, broadcast receiver, service, and content provider are essentials components. An Android application consists of one or more activities. An activity usually presents a single visual user interface in android. It’s like window or frame of Java. Service is a particular type of activity without a visual user interface and generally run in the background for an indefinite period of time [6].

Among the essentials components, activity and service encapsulate the most part of the behavior code. Android applications must adhere to strict life cycle protocols. An android activity is the subclass of ContextThemeWrapper class and controlled by 7 methods of android.app.Activity class. The 7 lifecycle method of Android Activity represent how activity will behave at different states. Figure 4 presents the 7 lifecycle methods of android activity. The onCreate method called when activity is first created. The onStart method is called when an activity is becoming visible to the user, while the onResume method is invoked when an activity will start interacting with the user, and the onPause method is called when an activity is not visible to the user. The onStop method called when activity is no longer visible to the user, so the onStart method is invoked after the stopped activity, prior to start, and onDestroy is used before the activity is destroyed [6].
5. MDA Approach for Android Applications.

This section presents the MDA approach for Android application development, which supports UML modeling and automatic code generation. This paper proposes how model the structure and behavior of an Android application using UML class diagram. Our modeling is based on Model View controller (MVC) pattern for implementing the android application.

The MVC pattern divides a software application into three interconnected parts: Models, Views, and Controllers. A model represents the raw information or access points to the raw information that is retrieved according to commands from the controller and displayed in the view. A view generates the visual representation of the models to the user based on changes in the model. And, the controller links the views to the models. Each event that is intercepted, as a touch, is sent to its respective controller. The controller then determines, according the input, which models to query. A controller sends the information collected to the model to update the model state. It may also send this information to its associated view to change the view’s presentation of the model (see Figure 5).
The main advantage of MDA in the development of Android application is the automation. This way, to demonstrate the automation support provides by our MDA approach, we are using the “translationist” approach [22]. With the translationist approach, the PIM is translated directly into the final code of the system by code generation. The translation of the PIM into final code is produced by a sophisticated code generator, sometimes named a ‘Model Compiler’, and symbolized by the large arrow. It is driven by ‘Generation Rules’ which describe how the elements of the PIM must be represented in the final code. The PSM is an intermediate step in the generation and is inside to the code generator. It is usually not visible or editable by the developer; it is shown with a dashed outline.

And the Acceleo [5] tool is used to generate the Android code. Our Android code generation is based on class diagrams, which are used by Acceleo to generate the application structure. The relationship between interfaces or classes, as association and inheritance, are respected during the code generation.

When the class represents an Android API component, the generation eventually includes necessaries imports, according to the attributes declaration and parameters of
methods. Moreover, the code generation provides invocation for Activity and Services standard methods, as `onCreate()`.

6. CASE STUDY: ERP Mobile Applications

Due to the increasing pressure on IT and standardization and cost, a growing number of enterprises have turned to Enterprise Resource Planning (ERP) systems to build their core IT system. The phenomenon is not limited to big companies, but the Small and Medium Industries (SMI) are reaching. Companies are setting to build responsive and prompt functions at all levels. An ERP system integrates all factors of business activities like human resources, sales and distribution, and finance and accounting, Finance and controlling … and on the other hand, Customer Relationship Management (CRM) systems is used to manage and track the customer relationship. Now these days, the next generation business is transitioning to commerce on mobile devices and smart phones.

Our case study adopts a single pattern: the CRUD-pattern, which allows creating, reading, updating and deleting instances of some entity. This pattern exists in many applications today; Figure 7 illustrates an example CRUD-pattern for a customer management application on an android smartphone. It shows a list of customers, where you can also create new customer, update existing ones, or delete customers that are no longer needed.

![Figure 7. Examples of Screens involved with a Classic CRUD-Pattern for an Entity](image)

We present here the different metaclasses forming the UML source metamodel used to develop the algorithm of transformation between the source and target model to generate the android code source. Figure 8 shows the source metamodel simplified UML model and the Figure 9 illustrates an Instance of UML Mode.
Figure 8. Simplified UML Source Metamodel

In this case, our application is constituted by 2 screens; first screen will display all customers in the database and the second screen is detail screen. We may automatically generate the source files in android project from a UML model consistent with our approach. The transformation from the PIM towards the android code source is realized with Acceleo, and the writing of the transformation rules itself does not present any problems in practice. It simply boils down to creating a text file where the transformation rules are written. Figure 10 presents the transformation rules to generate the AndroidManifest file. The manifest file presents fundamental information concerning
your application to the Android system, information the system must have before it can run any of the app’s code.

```
[template public pack2Manifest(aPack : Package)]
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.hanane. [aPack.name/]"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">

        [for (c:Class|aPack.elements)]
            <activity
                android:name=".MainActivity[c.name.toUpperFirst()]
                android:label="@string/app_name">

                [if (c.kind='main')]
                    <intent-filter>
                        <action android:name="android.intent.action.MAIN" />
                        <category android:name="android.intent.category.LAUNCHER" />
                    </intent-filter>
                [/if]

            </activity>
        [/for]

    </application>
</manifest>
[/template]
```

Figure 10. Extract of Transformation Code

7. Conclusion

Due to the big number and diversity of mobile technologies, the development of the same application for different platforms becomes a difficult and exhausting task. That’s because each platform uses different programming languages and tools, what makes difficult developing cross-platform applications. Thus, it requires software engineers to build applications over platforms, while ensuring the largest dissemination.
In this paper we propose a MDA approach for android applications development. This approach includes UML based modeling and automatic code generation using Acceleo with the aim to accelerate and facilitate the development of android applications. For future studies we will complete this code generator to automate the generation of source code for all mobile platforms (iOS, Black Berry, Windows Phone …).

References

Authors

Hanane BENOUDA, She got her Master Degree in Computer Engineering from Mohammed First university of Oujda, Morocco in 2013. She is currently a Ph.D Student in of Mathematics, signal and image processing and computing research Laboratory (MATS), EST Oujda, Morocco since 2014. Her main research interests include software engineering, mobile applications, reverse engineering …

Mostafa AZIZI, He received his diploma of State Engineer in Automation and Industrial Computing in 1993 from the Mohammadia School of engineers at Rabat (Morocco) and obtained his PH.D in Computer Science in 2001 from the Université de Montréal (DIRO-FAS) at Montreal (Canada). He is currently professor at the University of Oujda (Morocco). He teaches several courses in the domain of computer science such as OOP, IA, RT-systems, Distributed Systems, TCP/IP, WEB, Computers Security, Systems and Networking, Data structures and Algorithmic. He also supervises a number of Master/PH.D students. His research interests include: Verification/Coverification of real-time and embedded systems, Data communication and security, and Computer-aided management of industrial processes. An updated list of our publications is available on this link: scholar.google.com/citations?user=qlTcK5MAAAAJ

Mimoun MOUSSAOUI received his doctorate in Numerical Analysis in 1984 from University Paris XI (Orsay, France). He obtained his Ph. D in Nonlinear Analysis in 1991 from the Free University (Université Libre) of Brussels Belgium. He is currently professor at the University Mohamed First of Oujda (Morocco). He teaches Mathematics for economics and scientists, Numerical analysis and linear programming. He supervised several theses in applied mathematics and computing. He is director of Mathematics, signal and image processing and computing research Laboratory (MATS).

Redouane Esbai, He teaches the concept of Information System at Mohammed 1 University, He got his thesis of national doctorate in 2012. He got a degree of an engineer in Computer Sciences from the National School of Applied Sciences at Oujda. His activities of research in the MATSI Laboratory focusing on MDA (Model Driven Architecture) integrating new technologies XML, Spring, Struts, GWT, etc.