Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Licence en Informatique

Thème

Génération automatique d'une application Android

Réalisé par :

- **GHERNAOUT Malik**
- **AMAR Mourad**

Présenté le 28 Mai 2015 devant la commission d'examination composée de MM.

- ***HADJILA Fethallah*** *(Examinateur)*
- ***MERZOUG Mohamed*** *(Examinateur)*
- ***SMAHI Mohammed Ismail*** *(Encadreur)*

Année universitaire : 2014-2015

Remerciements

On remercie dieu le tout puissant de nous avoir donné la santé et la volonté d'entamer et de terminer ce mémoire.

Tout d'abord, ce travail ne serait pas le même sans l'aide et l'encadrement de Monsieur SMAHI Mohammed Ismail, on le remercie pour la qualité de son encadrement exceptionnel, pour sa patience, sa rigueur et sa disponibilité durant notre préparation de ce mémoire.

On remercie aussi les membres du jury Monsieur Hadjila Fethallah et Monsieur Merzoug mohamed d'avoir bien voulu examiner notre travail.

Enfin Nos profonds remerciements vont également à toutes les personnes qui nous ont aidés et soutenue de prés ou de loin.

Sommaire

Chapitre I

Présentation d'Android

I. 1 Introduction

Android est un système d'exploitation mobile utilisé dans les Smartphones, tablettes tactiles, assistants numériques personnels(PDA) mais aussi dans les téléviseurs, des radio-réveils, des montres connectées, des autoradios et même dans des ordinateurs de bords des voitures. Ce système d'exploitation qui utilise le noyau linux a été lance par une startup qui portait déjà le nom d'Android avant d'être achetée par Google en 2005. Ce projet est né d'un consortium de plus de 50 entreprises dont Google (initiateur du projet en novembre 2007) des fabriquant de matériels des operateurs mobiles des développeurs d'appliction, il est appelé l'Open Handset Alliance ou OHA cette figure représente les sociétés membres de l'Open Handset Alliance :

Figure I-1 les sociétés membre de l'OHA

L'objectif majeur était de développer des normes ouvertes pour des applications de téléphonie mobile et surtout à trouver une solution pour concurrencer l'IOS d'Apple, Windows Mobile de Microsoft, Symbian de Nokia et Research In Motion de Blackberry OS, en effet à cette époque la quasi-totalité des smartphones fonctionnaient sur ces systèmes d'exploitation. Cependant la principale différence entre Android et les autres solutions est qu'il soit open source c'est-à-dire un logiciel dans lequel le code source est à la disposition du grand public, et c'est généralement un effort de collaboration où les

programmeurs améliorent ensemble le code source et partagent les changements au sein de la communauté ainsi que d'autres membres peuvent contribuer. Cette particularité le rend donc gratuit et personnalisable par les constructeurs et les opérateurs de téléphonie mobile. Malgré des personnalisations accrues en terme d'interface utilisateur et de fonctionnalité le cœur du système reste commun ce qui permet une interopérabilité des applications. De ce fait, si l'on ne prend pas en compte les difficultés liées aux différences matérielles des périphériques, un téléphone Samsung et un autre Sony- Ericsson fonctionnant tous les deux sur Android seront en mesure d'exécuter les mêmes applications sans aucune compilation supplémentaire. Ce schéma représente le fonctionnement global de ce système d'exploitation on peut voire dans cette figure l'architecture Android.

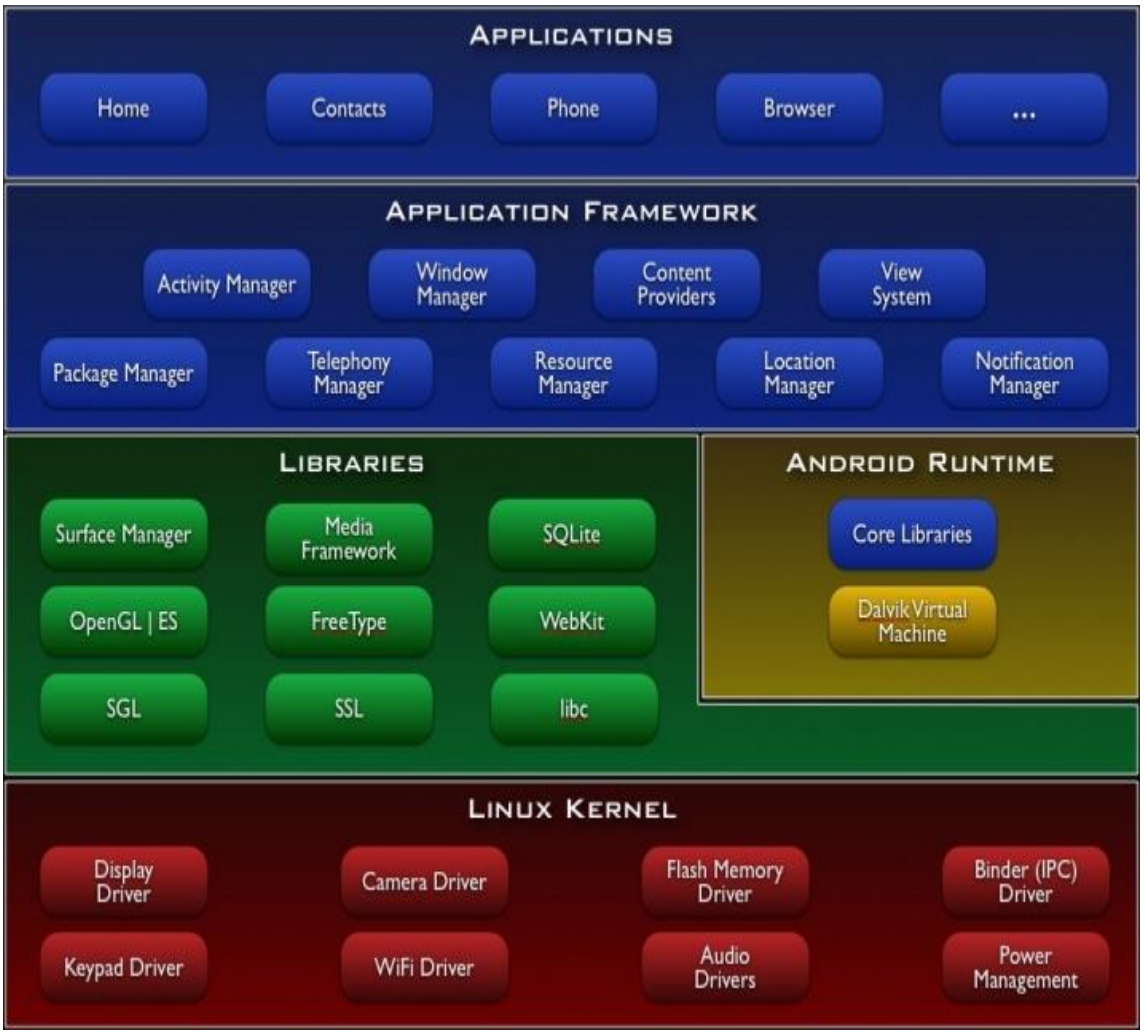

Figure I-2 l'architecture Android [3]

On peut y observer toute une pile de composants qui constituent le système d'exploitation. Le sens de lecture se fait de bas en haut, puisque le composant de plus bas niveau (le plus éloigné des utilisateurs) est le noyau Linux et celui de plus haut niveau (le plus proche des utilisateurs) est constitué par les applications.

Android est basé sur un kernel linux 2.6 mais ce n'est pas linux. Il ne possède pas de système de fenêtrage natif (X window system). La glibc n'étant pas supportée, Android utilise une libc customisée appelée Bionic libc. Enfin, Android utilise un kernel avec différents patches pour la gestion de l'alimentation, le partage mémoire, etc. permettant une meilleure gestion de ces caractéristiques pour les appareils mobiles. Android n'est pas linux mais il est basé sur un kernel linux.

Pourquoi sur un kernel linux ?

Le kernel linux a un système de gestion mémoire et de processus reconnu pour sa stabilité et ses performances. Le modèle de sécurité utilisé par linux, basé sur un système de permission, est connu pour être robuste et performant.

- Le kernel linux fournit un système de driver permettant une abstraction avec le matériel. Il permet également le partage de librairies entre différents processus, le chargement et le déchargement de modules à chaud.
- le kernel linux est entièrement open source et il y a une communauté de développeurs qui l'améliorèrent et rajoutent des drivers.

C'est pour les points cités ci-dessus que l'équipe en charge du noyau a décidé d'utiliser un kernel linux. Au-dessus de cette couche, on retrouve les librairies C/C++ utilisées par un certain nombre de composants du système Android. Au-dessus des librairies, on retrouve l'Android Runtime. Cette couche contient les librairies cœurs du Framework ainsi que la machine virtuelle Dalvik exécutant les applications. Au-dessus de la couche "Android Runtime" et des librairies cœurs, on retrouve le Framework permettant au développeur de créer des applications. Enfin au-dessus du Framework, il y a les applications.

2. Historique d'Android

En juillet 2005, Google a acquis Android Inc., une petite startup qui développait des applications pour téléphones mobiles .C'est à ce moment là que des rumeurs sur l'entrée de Google dans le secteur du mobile ont commencé. Mais personne n’avait des données sûres à propos des marchés dans lesquels ils allaient se positionner.

Après ce rachat fait par Google, une équipe dirigée par Andy Rubin, un ancien d'Android Inc., a commencé à travailler sur un système d'exploitation pour appareil mobile basé sur linux. Durant 2 ans, avant que l'OHA soit crée officiellement, un certain nombre de rumeurs ont circulé au sujet de Google. Il a été dit que Google développait des applications mobiles de son moteur de recherche, qu'elle développait un nouveau téléphone mobile, etc. En 2007, le 5 novembre, l'OHA a été officiellement annoncée, ainsi que son but: développer des standards open sources pour appareil mobile. Le premier standard annoncé a été Android, une plateforme pour appareils mobiles basée sur un kernel linux 2.6.

Figure I-3 différentes versions Android

En octobre 2008, apparait la première version d'Android qui n'avait pas reçu de nom. Cette version s'est avérée être la β du système.

La version 1.5 (Cupcake) corrigea le manque d'API et rendit le système plus utilisable.

Android 2.2 (Froyo) a fortement mis l'accent sur la synergie avec Internet. L'envoi d'applications et de liens instantanés depuis un ordinateur est désormais possible. Aussi, Google annonce que le navigateur chrome intégré à Android 2.2 est le navigateur mobile le plus rapide au monde grâce à l'intégration du moteur JavaScript V8.

Android 3.0(Honeycomb) est spécialement étudié pour les tablettes tactiles. Les premiers modèles sont sorties 2011.On y trouve quelques nouveautés comme la prise en charge de la vidéo-conférence via Gtalk, la nouvelle interface Gmail ou encore le lecteur

de livre électronique Google. La refonte graphique de l'interface utilisateur est assez réussie.

Android 4.0(IceCream) est arrivé très vite après le 3.0 (mi 2011) pour rajouter encore plus de fonctionnalités aux terminaux. Pour le développement, ces nouvelles versions d'Android proposent de nouveaux composants permettant de réaliser des applications avec une ergonomie plus adaptée aux tablettes tactiles.

Android 4.1 (Jelly Bean) sortie début septembre 2012, Il ajoute un système de notification améliorée, la reconnaissance vocale sans connexion internet, et le « Project Butter » qui augmente la fluidité d'Android .deux autres sous-versions sont sorties avant le lancement d'une nouvelle (KitKat).

Android 4.4 (KitKat) version sortie début octobre 2013, parmi les améliorations remarquées une Consommation en ressource moins élevée nécessitant moins de RAM, nouvelles icônes plus soignées, la barre du bas et celle de statut deviennent transparentes sur certains menus et changent de couleur en fonction du contenu affiché.

Android 5.0 (Lolipop) dernière version de ce système d'exploitation

Sortie novembre 2014 révisé en mars 2015 parmi les nouveautés :

- Android tv un dispositif de télévision connectée.
- Android auto qui permet d'interagir directement avec le système multimédia du véhicule
- Une Protection par blocage en cas de perte ou vol.

I.2 Les principales applications

Le système Android tout seul ne rend pas l'appareil réellement utilisable. Il fonctionne donc étroitement avec plusieurs applications, provenant de Google ou d'autres éditeurs et permettant une utilisation complète d'un smartphone. Ce type d'architecture donne aussi la possibilité aux développeurs et le choix à l'utilisateur final de remplacer les applications de base de leur appareil. Parmi ces applications, il y a notamment :

I.2.1 Les notifications

Une notification est une indication qui s'affiche sur la barre qui se situe en haut d'un téléphone Android.

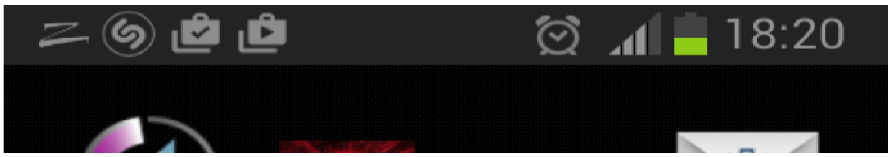

Figure I-4 Notification sur la barre des tâches

Cette notification sert à prévenir un utilisateur de certains évènements, comme la réception d'un message par exemple. Lorsque l'utilisateur glisse son doigt sur cette barre afin de la faire coulisser vers le bas, elle affiche les détails des notifications et permet, lors d'une pression sur ces notifications l'ouverture de l'application concernée.

I.2.2 Le home

Le home est l'application principale du système. Elle peut s'apparenter ordinateur. L'équipe de Google ayant développé le système a aussi développé un home sous licence open source. Ce home est représenté par trois bureaux coulissants de gauche à droite sur lesquels il est possible d'y placer des raccourcis vers les applications installées et des widgets, sorte d'interface minimaliste d'application.au bureau d'un système d'exploitation pour smartphone.

Figure I-5 Le home

I.2.3 Les applications téléphoniques

Sur Android, les applications pour toutes les utilités téléphoniques sont aussi remplaçables. C'est aussi vrai pour le répertoire des contacts, que pour le dialer ou la gestion des SMS et MMS. Ainsi, il existe plusieurs applications gratuites pour remplacer l'application d'origine qui gère les SMS et les MMS. Ces applications fournissent plusieurs avantages comme une personnalisation accrue de l'apparence, la gestion des MMS intégrés dans les fils de discussions ou encore la possibilité de dicter le message et de le lire par TextToSpeech.

I.2.4 Play Store

Afin que les développeurs puissent mettre à disposition de manière simple et efficace leurs applications aux utilisateurs du monde entier, Google a développé l'Android Market. Cette application, similaire à l'AppStore d'Apple pour l'iPhone, est le portail d'application par défaut des téléphones Android. Elle n'est pas Open source, il n'est donc pas légal de vendre un appareil ou publier une version d'Android avec l'Android Market sans l'autorisation de Google. Le Market permet la mise à disposition d'applications gratuites ou payantes et de les parcourir par catégorie. Chaque application affiche une icône, un nom d'application, une description, des captures d'écrans et les commentaires des utilisateurs. Pour ce qui est du mode de paiement utilisé par Google pour l'Android Market, il s'agit sans trop de surprises de Google Checkout, un concurrent direct de Paypal pour le paiement en ligne.

I.2.5 Le clavier virtuel

Android offre à l'utilisateur le choix de son clavier tactile. C'est une très bonne chose puisque cela pousse les éditeurs à trouver des solutions innovantes de saisie. Alors que certains tentent d'améliorer le clavier virtuel en mettant l'accent sur la correction à l'aide du T9 ou sur la réactivité et la prise en charge de la multitouche d'autres créent de nouveaux concepts de saisie allant d'une simple modification de la disposition des touches à de nouveaux concepts de saisis. C'est ainsi que la société Swype Inc. a créé une méthode basée sur le glissement du doigt sur chaque lettre composant un mot. Tout comme le reste des applications, l'installation d'un clavier virtuel se fera via un market et il sera possible de choisir quel clavier utiliser.

I.3 Le marché

I. 3.1 Évolution du marché des smartphones

Bien qu'Android soit utilisé pour une multitude de types d'appareil, du smartphone au netbook, nous allons nous concentrer sur le marché des smartphones puisque c'est sa principale cible

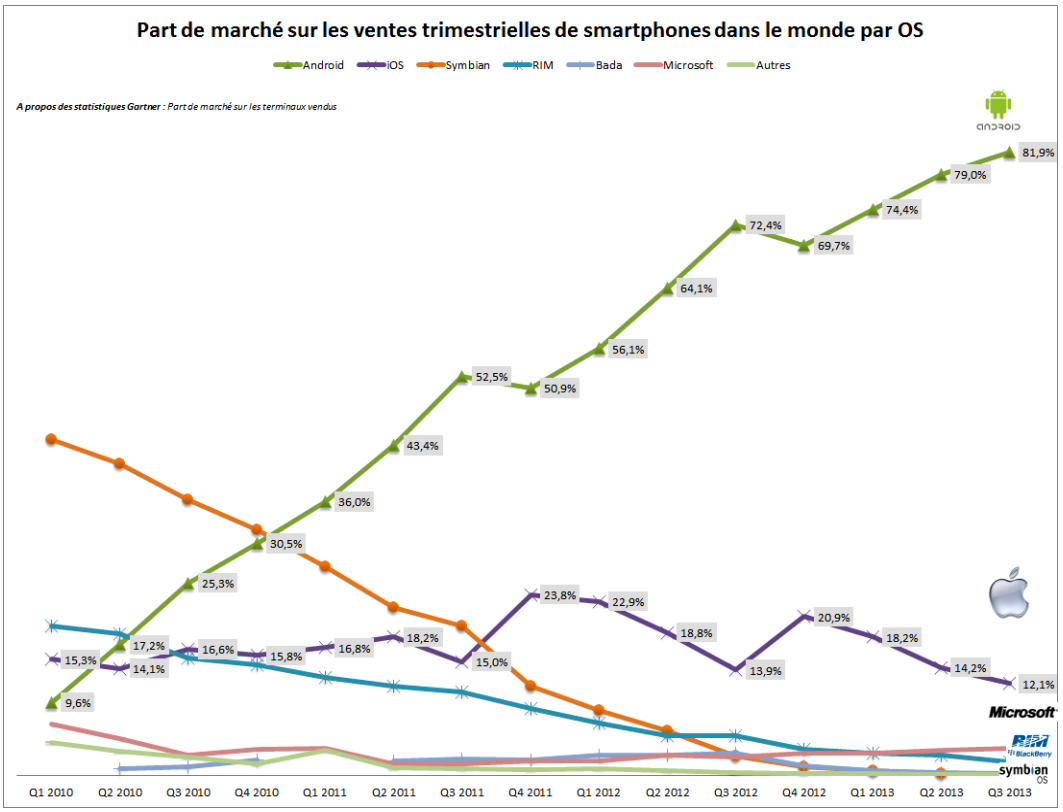

Figure I-3 graphe comparatif entre android et les systèmes concurrents

Cette courbe se passe pratiquement de tout commentaire et on voit qu'Android a largement Pris le dessus sur les autres systèmes d'exploitation. Aussi si Android atteint 79% de la part du marché c'est parce que Ce système d'exploitation alimente bon nombre de marques de smartphone et d'équipements électroniques contrairement par exemples a Ios d'Apple ou Rim de blackberry .

I. 3.2 Un même système d'exploitation pour une multitude d'appareils

Android étant gratuit et open source, il offre une personnalisation accrue, tout en profitant de la fiabilité d'un système commun et d'une interopérabilité des applications. De part ces particularités, les applications développées pour Android fonctionnent sur tous les appareils embarquant Android. Cependant, pour un bon fonctionnement, il est nécessaire de respecter certaines règles de programmation comme, par exemple, prendre en compte la résolution des écrans qui peut différer d'un appareil à l'autre. Il sera donc judicieux d'utiliser des unités de mesure adaptatives telles que des pourcentages plutôt que des pixels. Une autre différence d'un appareil à l'autre peut être la version d'Android, bien que les évolutions des versions d'Android apportent essentiellement des

améliorations du SDK, il est judicieux de les développer dans la mesure du possible pour la plus petite version d'Android.

Version	Codename	Nom API
2.2	Froyo	8
2.3.3 2.3.7	Gingerbread	10
3.2	IceCream	13
4.2. x 4.3	Jelly bean	17 18
4.4	KitKat	19
5.0 5.1. x	lollipop	21

Figure I-4 répartition des différentes versions android dans le marché [1]

Nous pourrions penser que tous les appareils vendus sous Android sont mis à jour avec les nouvelles versions du système, mais ce n'est pas le cas, les constructeurs doivent adapter leurs personnalisations car leurs drivers ne proposent pas la mise à niveau de toutes leurs gammes. Grâce à la mise sur le marché de nouveaux appareils et à la mise à jour de certains existants, cette répartition est amenée à évoluer. Il sera donc judicieux de consulter ce graphique avant de commencer tout projet de développement.

I.4 Remuneration

Afin de rentabiliser les applications, trois possibilités s'offrent à l'éditeur, nous allons donc les énumérer et citer leurs avantages et inconvenants.

I.4.1 Rémunération fixe

La rémunération fixe est la plus évidente, elle consiste à fixer un prix pour l'obtention de l'application, cependant plusieurs méthodes s'offrent à l'éditeur pour vendre son application. L'avantage de La rémunération fixe, c'est que ce n'est pas l'application qui

gère la transaction, mais bien Google. Le désavantage, c'est que Google garde 30% des revenus dans notre pays par exemple l'Ostore de Ooredoo conserve 50% du prix.

Bien entendu, ce type de paiement n'est pas adapté à toutes les applications. Il fonctionne très bien dans les jeux, mais n'est pas intéressant pour les applications professionnelles. La méthode la plus simple consiste simplement à diffuser l'application sur les différents market en y spécifiant son prix, cependant en plus de réduire la cible aux différents pays auxquels le market en question donne accès aux applications payantes, un pourcentage sera prélevé sur les ventes par la société fournissant le market.

Une autre solution serait alors de fournir une version de démonstration sur les différents market que l'utilisateur pourrait débloquer à l'aide d'un numéro de série qu'il aurait préalablement acheté sur un site de e-commerce. Cette solution, bien que plus contraignante pour l'utilisateur final à l'avantage d'éliminer la diffusion de l'application. D'autant que pour éviter l'utilisation de générateur de clé, cette clé peut être générer au moment de l'achat, stocker dans une base de données et lier à un compte client créé pour l'achat au niveau de l'application il suffit alors de demander en plus du numéro de série les identifiants de l'utilisateur.

I.4.2 Rémunération par la publicité

Une solution qui à l'avantage d'élargir sensiblement sa cible et de rémunérer l'application par la publicité plutôt que par un coût d'achat. Bien que la plupart des gens aient horreurs de la publicité, nous l'acceptons tous les jours par tous les moyens. Un bandeau de publicité a certainement moins de chance de repousser un utilisateur qu'un prix à payer, aussi minime soit-il, parmi les régies publicitaires proposant actuellement de la publicité pour les applications Android, il existe AdMob ou encore AdSense de Google.

I.4.3 Rémunération par abonnement

Pour certains services, il peut être souhaitable de distribuer l'application gratuitement, mais de demander un abonnement à l'utilisateur pour utiliser le service. Il est clair que ceci ne peut avoir d'intérêt que pour le cas d'application interfaçant un service web. Parmi les applications fonctionnant sur ce principe il y a Deezer et Spotify qui propose tous deux l'accès à leurs musiques en streaming contre un abonnement. Pour les personnes utilisant beaucoup leur smartphone comme lecteur de musique, cette fonctionnalité peutLeur être très avantageuse.

Conclusion

Dans ce chapitre nous avons donné tout d'abord une introduction sur ce système d'exploitation nous avons aussi vu l'architecture Android .Enfin nous avons présenté les différents modes de payement et rémunération proposées par google.

Dans l'optique de la réalisation d'une application android il existe de nombreuses approches qu'un développeur doit analysées dans le deuxième chapitre nous allons voire l'une de ces approches dite dirigées par les modèles.

Chapitre II

L'ingénierie dirigée par les modèles

Dans cette section nous définissons les principes fondamentaux de l'Ingénierie Dirigée par les Modèles notée IDM. Nous décrivons l'approche MDA (Model Driven Architecture) offerte par l'OMG (Object Management Group), qui constitue un espace technologique et un ensemble de standards pour l'application de l'IDM. Dans le but est d'introduire l'approche MDA dans un flot de conception des systèmes embarqués (constitués d'une partie matérielle et d'une partie logicielle).

L'ingénierie dirigée par les modèles (IDM), ou Model Driven Engineering (MDE) en anglais, a permis plusieurs améliorations significatives dans le développement de systèmes complexes en permettant de se concentrer sur une préoccupation plus abstraite que la programmation classique.il s'agit d'une forme d'ingénierie générative dans laquelle tout ou partie d'une application est engendrée à partir de modèles. Un modèle est une abstraction, une simplification d'un système qui est suffisante pour comprendre le système modélisé et répondre aux questions que l'on se pose sur lui. Un système peut être décrit par différents modèles liés les uns aux autres. L'idée phare est d'utiliser autant de langages de modélisation différents (Domain Specific Modeling Languages DSML) que les aspects chronologiques ou technologiques du développement du système le nécessitent. La définition de ces DSML, appelée métamodélisation, est donc une problématique clé de cette nouvelle ingénierie. Par ailleurs, afin de rendre opérationnels les modèles (pour la génération de code, de documentation et de test, la validation, la vérification, l'exécution, etc.), une autre problématique clé est celle de la transformation de modèle.

II. 1 Les principes généraux de l'IDM

Suite à l'approche objet des années 80 et de son principe du « tout est objet », l'ingénierie du logiciel s'oriente aujourd'hui vers l'ingénierie dirigée par les modèles (IDM) et le principe du « tout est modèle ». Cette nouvelle approche peut être considérée à la fois en continuité et en rupture avec les précédents travaux. Tout d'abord en continuité car c'est la technologie objet qui a déclenché l'évolution vers les modèles. En effet, une fois acquise la conception des systèmes informatiques sous la forme d'objets communicant entre eux, il s'est posé la question de les classifier en fonction de leurs différentes origines (objets métiers, techniques, etc.). L'IDM vise donc, de manière plus radicale que pouvaient l'être les approches des patterns et des aspects, à fournir un grand nombre de modèles pour exprimer séparément chacune des préoccupations des

utilisateurs, des concepteurs, des architectes, etc. C'est par ce principe de base fondamentalement différent que l'IDM peut être considérée en rupture par rapport aux travaux de l'approche objet. Alors que l'approche objet est fondée sur deux relations essentielles, « Instance De » et « Hérite De », l'IDM est basée sur un autre jeu de concepts et de relations. Le concept central de l'IDM est la notion de modèle. Cette figure ci-dessous montre les relations de base de l'IDM «conforme a» «représentation de».

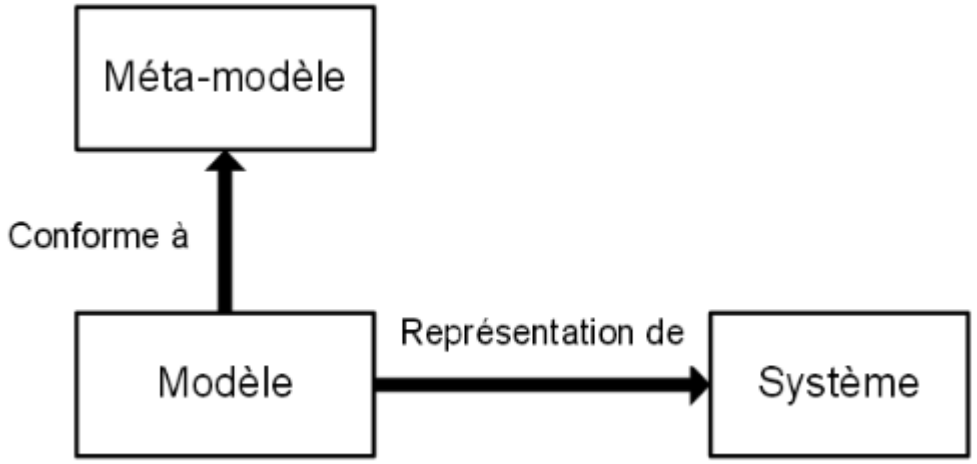

Figure II.1 Relations de base de l'IDM [6]

II.1.1 Qu'es qu'un modèle :

Bien que possédant un grand nombre de définitions, un modèle dans l'IDM ('Ingénierie dirigée par les modèles) est l'abstraction d'un objet sujet de l'acte de modélisation. Son but est de permettre une étude plus simple dans un contexte maitrisé autre que le contexte réel. Le modèle doit pouvoir être utilisé pour répondre à des questions sur le système modélisé. Aussi un modèle a pour objectif de structurer les données, les traitements, et les flux d'informations entre entités. L'objectif du processus de l'ingénierie de conception est de rendre notre monde mesurable, calculable, prévisible et donc plus facile à gérer.

Avantages d'un modèle

Abstrait

- Il fait ressortir les points importants tout en enlevant les détails non nécessaires

Compréhensible

- Il permet d'exprimer une chose complexe dans une forme plus facilement compréhensible par l'observateur

Précis

- Il représente fidèlement le système modélisé

Prédictif

- Il permet de faire des prévisions correctes sur le système modélisé

Peu coûteux

- Il est bien moins coûteux à construire et étudier que le système lui même

La notion de modèle dans l'IDM fait explicitement référence à la notion de langage bien défini. En effet, pour qu'un modèle soit productif, il doit pouvoir être manipulé par une machine. Le langage dans lequel ce modèle est exprimé doit donc être clairement défini. De manière naturelle, la définition d'un langage de modélisation a pris la forme d'un modèle, appelé métamodèle.

II.1.2 Qu'es qu'un Métamodèle

Un métamodèle est un modèle qui définit le langage d'expression d'un modèle, c'est-à-dire le langage de modélisation. La notion de métamodèle conduit à l'identification d'une seconde relation, liant le modèle et le langage utilisé pour le construire. La métamodélisation est l'activité consistant à définir le métamodèle d'un langage de modélisation. Elle vise donc à bien modéliser un langage, qui joue alors le rôle de système à modéliser.

II.1.3 Qu'est-ce qu'un langage

Que ce soit en linguistique (langage naturel) ou en informatique (langage de programmation ou de modélisation), il est depuis longtemps établi qu'un langage est caractérisé par sa syntaxe et sa sémantique. La syntaxe décrit les différentes constructions du langage et les règles d'agencement de ces constructions. La sémantique désigne le lien entre un signifiant (un programme, un modèle, etc.), et un signifié (un objet).

II.2 Outils et techniques pour la spécification d'un langage de modélisation

II.2.1 Syntaxe abstraite

La syntaxe abstraite (AS) d'un langage de modélisation exprime, de manière structurelle, l'ensemble de ses concepts et leurs relations. Les langages de métamodélisation tels que le standard MOF, offrent les concepts et les relations élémentaires qui permettent de décrire un métamodèle représentant la syntaxe abstraite d'un langage de modélisation. Pour définir cette syntaxe, nous disposons à ce jour de nombreux environnements et langages de métamodélisation. Parmi eux Eclipse-EMF ou Kermeta.

II.2.2 Syntaxe concrète

Les syntaxes concrètes (CS) d'un langage fournissent à l'utilisateur un ou plusieurs formalismes, graphiques et/ou textuels, pour manipuler les concepts de la syntaxe abstraite et

ainsi en créer des « instances ». Le modèle ainsi obtenu sera conforme à la structure définie par la syntaxe abstraite.

II.3 L'architecture dirigée par les modèles (MDA)

L'architecture dirigée par les modèles ou MDA (Model Driven Architecture) est une démarche de réalisation de logiciels, proposée et soutenue par l'OMG (Object management group). C'est une variante particulière de l'ingénierie dirigée par les modèles (IDM). D'autres variantes de l'IDM ont été développées, par exemple par Microsoft (DSL Tools).Le MDA est une initiative de l'OMG rendue publique en 2000. C'est une proposition à la fois d'une architecture et d'une démarche de développement. L'idée de base du MDA est la séparation des spécifications fonctionnelles d'un système des détails de son implémentation sur une plate-forme donnée.

L'approche MDA distingue deux aspects principaux dans le processus de développement d'une application, l'aspect métier qui représente les fonctions de l'application, et l'aspect technique qui représente la technologie de mise en œuvre de l'application. Chaque aspect est exprimé par un ensemble de modèles, qui véhiculent l'information nécessaire à la génération du code source de l'application. On passe d'une vue contemplative des modèles à une vue productive. MDA définit trois niveaux de modèles représentant les niveaux d'abstraction de l'application, le CIM, le PIM et le PSM :

II.3.1Modèle CIM- Computational Independant Model

Les modèles d'exigence CIM décrivent les besoins fonctionnels de l'application, aussi bien les services qu'elle offre que les entités avec lesquelles elle interagit. Leur rôle est de décrire l'application indépendamment des détails liés à son implémentation. Les CIM peuvent servir de référence pour s'assurer que l'application finie correspond aux demandes des clients.

II.3.2 Modèle PIM- Platform Independant Model

Les modèles PIM sont les modèles d'analyse et de conception de l'application. La phase de conception à cette étape du processus suppose l'application de Design pattern, le découpage de l'application en modules et sous-modules, etc. Le rôle des PIM est de donner une vision structurelle et dynamique de l'application, toujours indépendamment de la conception technique de l'application.

II.3.3 Modèle PM- Platform Model

Rarement utilisé, un PM décrit la structure, et les fonctions techniques relatives à une plateforme d'exécution (systèmes de fichiers, de mémoire, de BDD…) et précise comment les utiliser. Le PM est associé au PIM pour obtenir le PSM.

II.3.4 Modèle PSM- Platform Specific Model

Le PSM est le modèle qui se rapproche le plus du code final de l'application. Un PSM est un modèle de code qui décrit l'implémentation d'une application sur une plateforme particulière, il est donc lié à une plateforme d'exécution.

II.3.5 Code source

Représente le résultat final du processus MDA, le code source est obtenu par génération automatique (partielle ou totale) du code de l'application à partir du PSM. Le code source obtenu peut toujours être enrichi ou modifié manuellement.

II.4 Les transformations des modèles

Les modèles CIM, PIM et PSM constituent les étapes principales de l'approche MDA. Chacun de ces modèles contient des informations nécessaires à la génération du code source de l'application. Le code est obtenu par génération automatique à partir du PSM, le PSM est obtenu par transformations successives des modèles CIM vers PIM et des modèles PIM vers PSM. La transformation de modèles est une étape importante du processus MDA, c'est grâce aux transformations que les modèles deviennent des éléments productifs de MDA. L'exécution des transformations permet d'assurer un lien de traçabilité entre les différents modèles du processus MDA. Ces liens sont un gage de qualité du processus de développement logiciel dans le MDA.

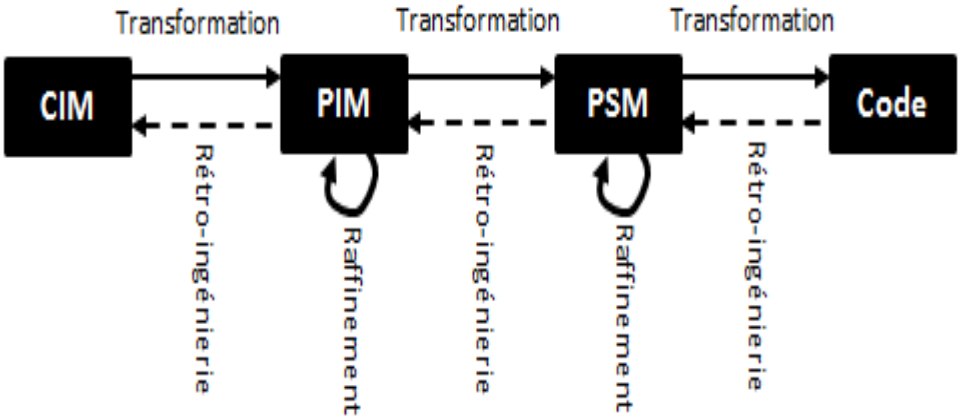

Figure II.2 transformations des modèles dans le MDA[6]

II.4 .1 Transformations CIM vers PIM

Les modèles CIM expriment les besoins des utilisateurs. Cette étape consiste à construire, partiellement, des modèles PIM à partir des CIM. Le but est de retranscrire les informations contenues dans les CIM vers les modèles PIM. C'est ce qui va permettre de s'assurer que les besoins de l'utilisateur sont véhiculés et respectés tout au long du processus MDA.

II.4 .2 Transformations PIM vers PIM

Les modèles PIM modélisent l'aspect structurel et dynamique d'une application. Cette étape s'exprime par l'enrichissement des modèles PIM. Enrichir des PIM consiste à leur rajouter de l'information utile et à spécifier leur contenu. Les PIM sont ainsi raffinés et les informations qu'ils contiennent, précisées.

II.4 .3 Transformations PIM vers PSM

Cette étape consiste à créer des modèles PSM à partir des informations fournies par les modèles PIM, en y ajoutant des informations techniques relatives à la plateforme d'exécution cible. C'est ainsi que le lien avec la plateforme d'exécution se forme. Un PSM fournit les informations utiles à la génération du code de l'application et est dépendant de la plateforme d'exécution. Il est possible de créer autant de PSM qu'il y a de plateformes cibles.

II.4 .4 Transformations PSM vers code

La transformation des modèles PSM en code source consiste à générer le code source de l'application, de façon totale ou partielle, à partir des modèles PSM de l'application. Cette étape n'est pas à proprement dit considérée comme une transformation par MDA. En effet, une transformation selon MDA est définie par la transformation d'un modèle vers un autre modèle, chacun d'eux étant structuré par leur métamodèle. Or le code source n'a pas de métamodèle, la transformation des PSM vers le code est plutôt considérée comme une retranscription textuelle du modèle PSM.

II.4 .5 Transformations inverses

Transformations inverses ou rétro-ingénierie, signifie la construction de modèles à partir d'applications existantes. Dans ce cadre, MDA identifie aussi les transformations inverses : Code vers PSM, PSM vers PIM et PIM vers CIM.

II.5 Développement des modèles

II.5.1 Les technologies mises en œuvre

L'approche MDA préconise l'utilisation de standards de l'OMG, parmi eux UML, MOF ET XMI, mais n'exclut pas non plus l'utilisation d'autres outils ou langages.

Pour l'élaboration des modèles, MDA préconise l'utilisation du standard UML. UML définit un ensemble de diagrammes permettant de modéliser tous les aspects d'une application orientée objet (aspect statique, dynamique, fonctionnel…). Pour chaque modèle MDA correspond un diagramme UML approprié.

Les profils UML présentent l'avantage de faciliter la transformation PIM vers PSM par rapport aux PM, mais ces derniers permettent une plus grande expressivité pour les plateformes.UML modélise une application orientée objet sous forme d'un ensemble d'objets liés par des relations, mais il ne permet pas de définir le comportement de ces objets. C'est pour corriger cette lacune que l'OMG a défini les standards OCL (Object Constraint Language) et AS (Action Semantics). UML décrit une opération d'un objet que par son nom, ses paramètres et les exceptions qu'elle lève, OCL et AS se greffent à UML pour ajouter plus de précision à cette description en offrant les outils pour permettre la définition du corps de ces opérations. Les standards OCL et AS sont indépendants des plateformes d'exécution, ils sont principalement utilisés pour l'élaboration des PIM.

II.6 Les transformations architecturale

L'approche MDA préconise de modéliser les transformations elles-mêmes. MDA considère une transformation comme une application. Un modèle de transformation est une fonction qui prend un ensemble de modèles en entrée et rend un ensemble de modèles en sortie.

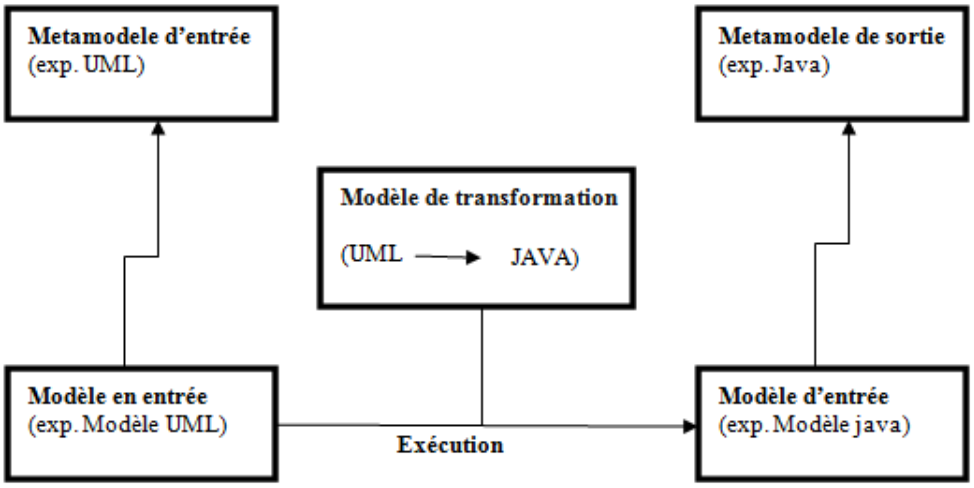

Figure II.3 Structure des modèles en entrée et en sortie

Les modèles, en entrée et en sortie, sont structurés par leur métamodèle.

Il existe trois façons différentes de modéliser une fonction de transformation. Le principe reste le même, la différence réside dans la façon de formuler les règles de transformations :

II.6.1 Approche par programmation

Dans cette approche, l'idée est de programmer une transformation de modèle de la même façon que n'importe quelle application informatique, en utilisant les langages orientés objet. Et où les données à manipuler représentent les modèles impliqués dans la transformation.

II.6.2 Approche par template

L'approche par template consiste à définir des modèles templates et à remplacer leurs paramètres par les valeurs des modèles sources. Les modèles templates, ou modèles cibles paramétrés, sont des canevas des modèles cibles, ils contiennent des paramètres qui seront par la suite, lors de la transformation, substitués par les valeurs des modèles sources. Cette approche utilise un langage particulier pour la définition des modèles templates.

II.6 .3Approche par modélisation

Par analogie à l'approche MDA, l'approche par modélisation modélise aussi les règles de transformations en se basant sur l'ingénierie dirigée par les modèles. Dans cette approche, un modèle de transformations est structuré par son métamodèle, et les modèles sont indépendants des plateformes d'exécution. L'objectif de cette approche est de pérenniser les modèles de transformations et de les rendre productifs. L'OMG a défini le standard MOF2.0 QVT (Query/View/Transformation) comme métamodèle pour structurer les modèles de transformations de modèles. Les modèles de transformations, conformes au métamodèle MOF2.0 QVT, expriment les règles de correspondance entre le métamodèle source et le métamodèle cible nécessaires à la transformation.

Objectifs du MDA

L’approche MDA pour répondre aux problèmes liés à l'évolution continue des technologies. MDA est à l'origine de l'ingénierie dirigée par les modèles. Tout ce qui constitue l'approche MDA, la modélisation, les transformations, les technologies de mises en œuvre a été pensé afin de permettre à MDA de remplir les objectifs suivants :

- **La pérennité des savoir-faire**

La technologie évolue plus vite que les métiers de l'entreprise. Il semble alors plus intéressant de capitaliser les savoir-faire de l'entreprise indépendamment des préoccupations techniques. L'approche MDA vise principalement à rendre les spécifications métier des entreprises pérennes, indépendamment des technologies de mise en œuvre. Cette pérennité est possible grâce aux modèles, qui sont par nature des entités pérennes, et à l'utilisation d'UML qui est un standard stable et largement répandu.

- **Les gains de productivité**

Les modèles sont au cœur du processus MDA, ils facilitent la communication entre experts du domaine et développeurs, mais pas seulement. Dans MDA les modèles deviennent un outil de production grâce à l'automatisation des transformations de modèles. Le modèle devient un élément qui véhicule une information utile, précise et nécessaire, permettant l'obtention du code source de l'application par génération automatique du code. L'automatisation des transformations que permettent les modèles dans le processus de développement MDA, entraîne un gain de productivité.

- **La prise en compte des plateformes d'exécution**

À partir du PIM, MDA permet d'intégrer les spécifications techniques d'une plateforme d'exécution dans les transformations PIM vers PSM, et d'obtenir ainsi un PSM spécifique à la plateforme d'exécution, à partir duquel le code source de l'application est généré. Grâce à ce procédé, une application tire pleinement parti des fonctionnalités de la plateforme d'exécution cible. Le développement d'applications multiplateformes, ou encore la migration logicielle, sont facilités puisqu'il suffit à partir du PIM, d'effectuer les transformations PIM vers PSM en y intégrant à chaque fois, le modèle de la plateforme concernée. Obtenant ainsi pour chaque plateforme, son PSM à partir duquel le code source sera généré.

Conclusion

Au cours de ce chapitre nous avons vu ce qu'est l'IDM et le MDA nous avons aussi présenté les transformations des modèles .nous avons aussi vu brièvement d'autres types d'approches notamment l'approche par templates, par programmation. Enfin nous avons conclu ce chapitre par donné quelques objectifs de cette architecture dirigées par les modèles.

Chapitre III

Réalisation de l'application ApkStore

III.1 Langages

III.1.1 Java

Pour développer sur Android, Google à mis à disposition un SDK² permettant d'accéder à toutes les fonctionnalités des appareils, par exemple l'accéléromètre ou l'écran tactile et ceci de manière simple en utilisant des librairies existantes, l'ensemble de ces librairies constitue le Framework Android. Ce SDK, basé sur du Java, demande quelques mises à niveau, mais est à la portée de tout développeur ayant des connaissances en développement orienté objet.

Pour des jeux ou des librairies demandant un maximum de ressources ou ayant besoin d'une optimisation accrue, un NDK3 est mis à disposition. Ce NDK permet de compiler du code C++ offrant une exécution beaucoup plus rapide qu'avec le Java.

III. 1.2 Le Java Développement Kit (JDK)

La machine virtuelle Java ou JVM (Java Virtual Machine) est un environnement d'exécution pour applications Java.

C'est un des éléments les plus importants de la plate-forme Java. Elle assure l'indépendance du matériel et du système d'exploitation lors de l'exécution des applications Java. Une application Java ne s'exécute pas directement dans le système d'exploitation mais dans une machine virtuelle qui s'exécute dans le système d'exploitation et propose une couche d'abstraction entre l'application Java et ce système.

La machine virtuelle permet notamment :

- l'interprétation du bytecode
- l'interaction avec le système d'exploitation
- la gestion de sa mémoire grâce au ramasse miettes (GarbageCollector)

III.1.3 Interface XML

Dans la lignée du Flex d'Adobe ou du XAML de Microsoft, les interfaces utilisateurs sous Android sont définies dans un langage balisé de type XML1. Cette approche donne la possibilité au développeur d'écrire son interface, d'utiliser un outil de création d'interface ou encore de le développer lui même. Ce format étant ouvert et humainement compréhensible, il est facilement traitable dans un script ou une application pour, par exemple, remplacer en masse un élément présent dans plusieurs interfaces.

III.2 kit de développement (SDK)

Le kit de développement (SDK) d'Android est un ensemble complet d'outils de développement1.Il inclut un débogueur , des bibliothèques logicielles ,un émulateur basé sur QEMU(logiciel libre de machine virtuelle), de la documentation, des exemples de code et des tutoriaux. Les plateformes de développement prises en charge par ce kit sont les distributions sous Noyau Linux, Mac OS, Windows XP ou version ultérieure. L'IDE officiellement supporté était Eclipse combiné au plugin d'outils de développement d'Android (ADT), mais depuis 2015, Google officialise Android Studio qui devient alors l'IDE officiel pour le SDK Android. Les développeurs peuvent utiliser n'importe quel éditeur de texte pour modifier les fichiers Java et XML, puis utiliser les outils en ligne de commande (Java Development Kit et Apache Ant sont obligatoires) pour créer, construire et déboguer des applications Android ainsi que contrôler des périphériques Android. Les possibilités offertes par le SDK regroupent, entre autres, l'accès à toutes les fonctionnalités de l'appareil comme l'accès à internet, à la partie téléphonie (appel, gestion des contacts et des SMS), à l'appareil photo, au GPS, à l'accéléromètre ou encore à la boussole

III.2.1 Les outils mis à disposition par le SDK

III.2.1.1 ADB (Android Debug Bridge)

ADB est un outil disponible en ligne de commande. Comme son nom l'indique, il permet de se connecter à l'émulateur ou au téléphone à l'aide d'un câble USB standard.En outre, il donne la possibilité d'installer ou de désinstaller des applications, d'envoyer ou de récupérer des fichiers et de se connecter en ligne de commande sur l'appareil. Il est aussi utilisé par le SDK pour publier et débugger les applications en développement et peut-être aussi sollicité par certaines applications, par exemple, « Remote SMS » qui permet d'envoyer des SMS depuis le navigateur d'un ordinateur.

III.2.1.2 L'émulateur

L'Android Virtual Device, aussi appelé AVD, est un émulateur de terminal sous Android, c'est-à-dire que c'est un logiciel qui se fait passer pour un appareil sous Android à l'ordinateur. C'est la raison pour laquelle vous on n'pas besoin d'un périphérique sous Android pour développer et tester la plupart des applications. En effet, une application qui affiche un calendrier par exemple peut très bien se tester dans un émulateur, mais une application qui exploite le GPS doit être éprouvée sur le terrain pour que l'on soit certain de son comportement.

Figure III.1 l'émulateur Android

III.3 L'environnement de travail (IDE)

L'environnement de travail sélectionné pour ce projet de création d'application android s'est présenter en deux choix soit android studio devenu depuis quelques mois l'environnement off soit éclipse ide nous avons choisit ce dernier car il a toujours été le produit phare du développement Android et Eclipse fait partie des grands logiciels libres réputés

III.3.1 présentation d'éclipse

Eclipse est un IDE, Integrated Development Environment (EDI environnement de développement intégré en français), c'est-à-dire un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation. Il est développé par IBM, est gratuit et disponible pour la plupart des systèmes d'exploitation. C'est est un environnement intégré de développement (IDE) pour le langage Java (et d'autres langages).

Au fur et à mesure que l'on programme, éclipse compile automatiquement le code qu'on écrit, en soulignant en rouge ou jaune les problème qu'il décèle. Il souligne en rouge les parties du programme qui ne compilent pas, et en jaune les parties qui compilent mais peuvent éventuellement poser problème (on dit qu'éclipse lève un avertissement, ou warning en anglais). Pendant l'écriture du code, cela peut sembler un peu déroutant au début, puisque tant que la ligne de code n'est pas terminée (jusqu'au point-virgule), éclipse indique une erreur dans le code.

Eclipse a été créé par l'OTI (Object Technology International) et les équipes d'IBM Initialement composé de 40 développeurs à plein temps.il peut être téléchargé sur le site www.eclipse.org.

*les points forts d'éclipse :

Eclipse ide présente des qualités indéniables qui ont faits et qui font encore son succès par ces qualités ont trouve notamment :

- Il est Prévu pour fournir une plateforme ouverte de développement:

En effet éclipse fonctionne sur un grand nombre de systèmes d'exploitation.et en plus son Interface graphique est très performante et facilitant le développement d'applications.

- Indépendance du langage de programmation :

Eclipse permet sans restriction l'utilisation plusieurs types de contenus.
HTML, Java, C, JSP, EJB, XML, GIF…

- Une facilité d'intégration de nouveaux outils :

Au niveau de l'interface et en profondeur.
Ajout de nouveaux outils pour les produits installés.

- Il attire une grande communauté de développeurs :

Y compris des éditeurs de logiciels indépendants.

Capitalise la popularité de Java pour l'écriture de nouveaux Outils.

III.3.2 Création d'un projet android sous éclipse

Apres l'instalation d'eclipse on lance "Eclipse.exe"

Dans Eclipse : File / New/ Project

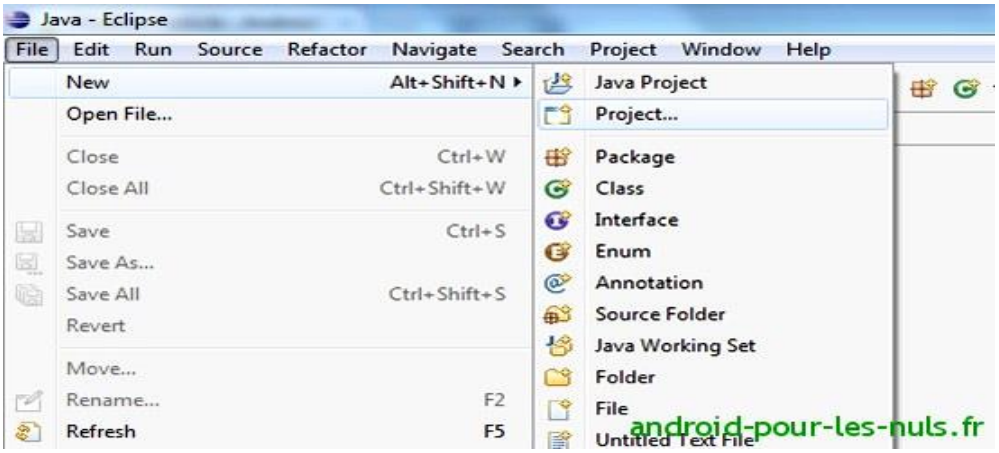

Figure III.2 étapes de création d'un projet éclipse [8]

Puis Android Project

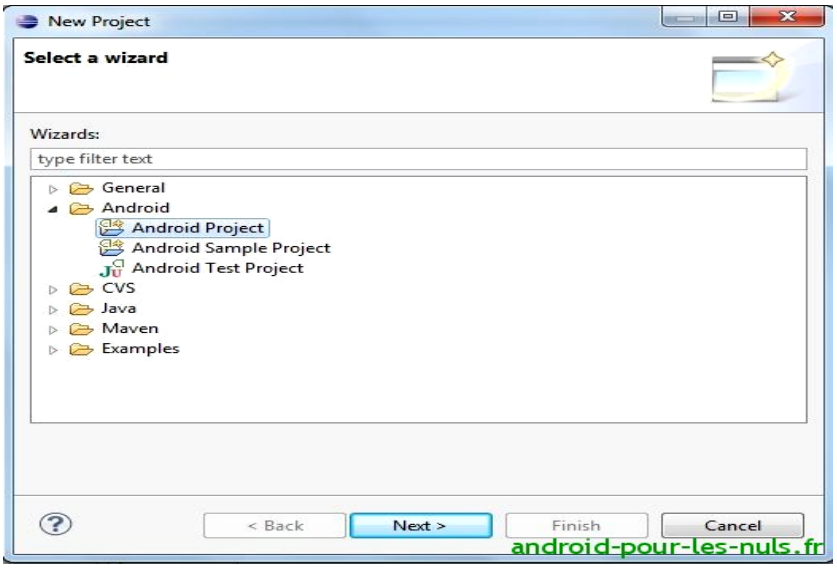

Figure III.3 étapes de création d'un projet éclipse[8]

On click en suite sur ''next'' pour donner le nom de l'application, on choisit la version android Ciblée par notre travail aussi on choisit la version minimal sous la quelle notre application Fonctionnera on clique en suite sur 'finish'.dés lors la fenêtre d'éclipse génère une multitude de fichier qu'il est a première vue difficile a connaitre, on va expliquer l'utilité de ces ficher et leurs rôle dans le fonctionnement d'une application android.

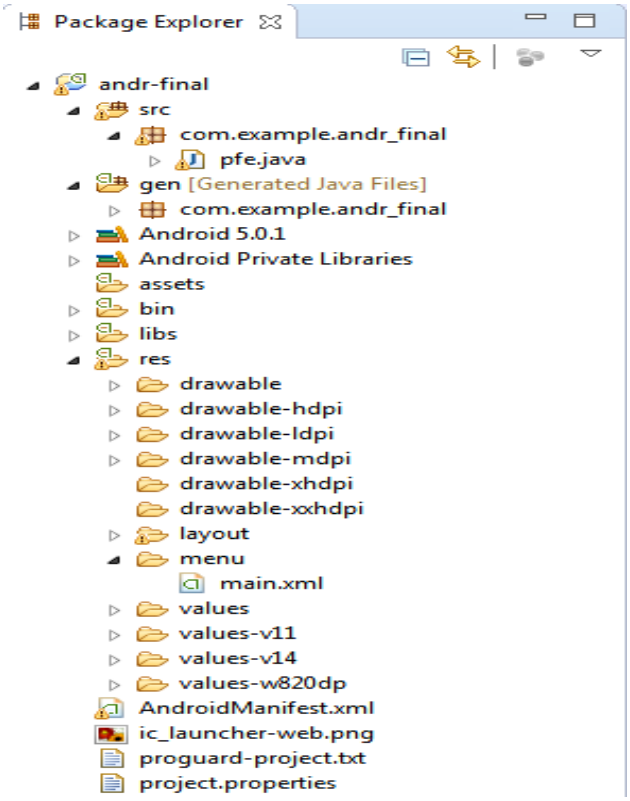

Figure III.4 étapes de création d'un projet éclipse[8]

• **src** contient des fichiers sources correspond aux sources Java du projet. C'est dans ce dossier src que sont disposées les « Activity », les classes d'accès aux données et toute la partie moteur du projet

• **gen** contient les fichiers générés par Eclipse comprend les fichiers générés par le SDK et les librairies qu'utilise le projet. Les fichiers générés ne doivent pas être modifiés puisqu'ils sont souvent mis à jour automatiquement par le SDK. Le fichier « R.java » permet au SDK de faire le lien entre les identifiants utilisés dans les sources et les ressources auquel ils correspondent

• **res** contient les ressources (images, descriptions d'interfaces, valeurs)

Contient les différentes ressources du programme. Il peut s'agir d'image, de son, de vue ou de n'importe quels fichiers devant être utilisés par l'application.

Plus précisément, le dossier « assets » permet la copie de fichiers de tous types alors que le dossier « res » correspond aux ressources identifiables par Le SDK. Les ressources contrairement aux « assets » sont prises en compte par le SDK. Elles doivent être classées dans des sous-dossiers spécifiques et seront accessibles dans le code de manière simplifiée.C'est dans le fichier « R.java » que le SDK se chargera de renseigner l'ajout ou la modification de ressources.

•**AndroidManifest** décrit les propriétés de l'application qui correspond au fichier définissant les propriétés de l'application et du projet. Le fichier « AndroidManifest .xml » comprend toutes les propriétés de l'application comme son nom, son « package name », ou encore sa version. C'est aussi dans ce fichier que l'on va spécifier les permissions demandées par l'application. Le fichier « default.properties » est un fichier générer automatiquement par Eclipse et contient les propriétés du projet. Ce fichier n'est donc pas à prendre en considération

III.4 Présentation de l'application

Le but de notre travail a été de créé trois applications de services android de messagerie, d'Envoi d'email et de géo localisation pour ensuite les héberger dans un serveur web est de télécharger les apks à l'aide de l'interface apkstore .la figure ci-dessus illustre l'interface de l'application

Figure III-5 interface Apk store

Cette interface nous donne le choix de sélectionner la version android Que l'on veut et qui doit être compatible avec la version du smartphone qui utilisera ces services.

Cette interface nous donne aussi la main afin de choisir le service désiré. La sélection se fait à l'aide de checkbox ces services qui sont :

- un service de messagerie.
- un service pour l'envoie d'email.

- un service de géo localisation.

En choisissant l'api et le service dont on a besoin et cliquant sur le bouton illustré par la figure ci-dessus on lance le téléchargement de l'apk depuis le serveur distant.

Figure III-6 bouton de téléchargement des apks

III.4.1 Les services utilisés :

- le service Email

Ce service nous offre l'opportunité d'envoyer des courriers électroniques d'une facon tres simple et ce en saisissant l'adresse electronique du correspondant, l'objet et le message, ces figues illustrent parfaitement le protocole à suivre afin d'envoyer le e-mail a partir de cette application

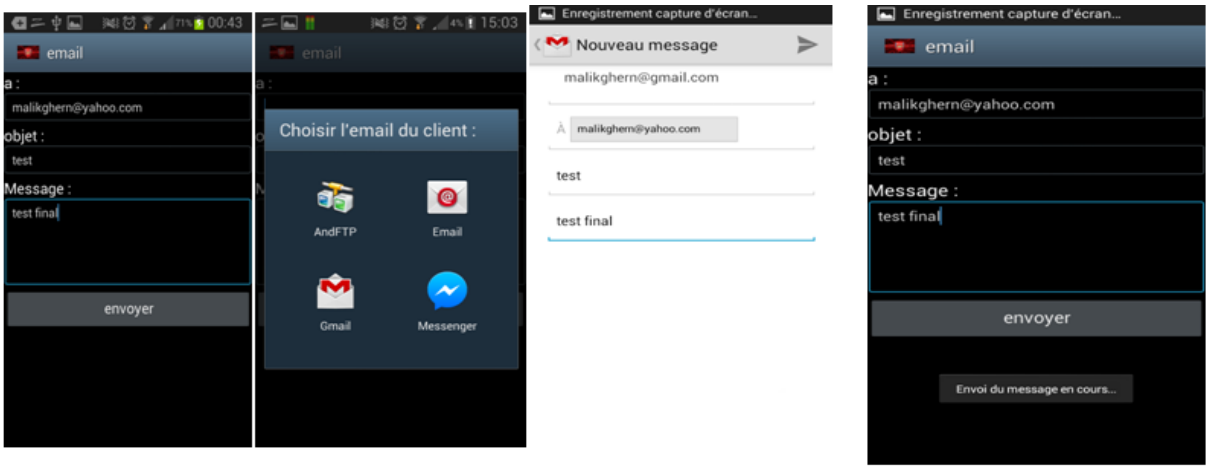

Figure III-7 étapes d'envoi d'un email

On remplis les trois champs ensuite on choisi par quelle voie on envoi notre email,

En choisissant gmail on va envoyer notre mail via ce service et on n'a plus qu'à sélectionner la flèche envoyé .

- Le service sms :

C'est un service messagerie simple pour envoyer des messages depuis le Smartphone l'interface d'utilisation de cette application se présente dans la figure ci-dessus

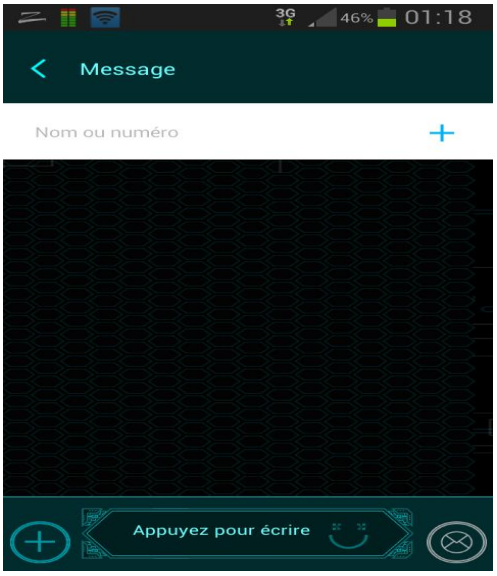

Figure III-8 interface de l'application de messagerie

Conclusion generale

Notre projet a consisté en la conception, le développement et l'intégration d'une application de téléchargement des applications Android « apkStore », afin d'apporter une valeur ajoutée et un meilleur service aux utilisateurs de l'application.

Nous sommes arrivés à développer toutes les fonctionnalités du système. L'intégration a été réalisée avec succès, c'est-à-dire que l'application est maintenant fonctionnelle installée sur tablette et Smartphone. Ce qui nous a permis d'acquérir des connaissances théoriques et la pratique de nouvelles technologies. En outre elle nous a permis de maîtriser le langage de programmation java, les outils de développement Android à savoir le SDK Android, sous lequel, le développement n'a pas été une tâche facile, mais nous n'avons pas hésité à relever le défi. Il nous a également permis de découvrir comment se passe l'intégration d'une application sur un serveur web distant pour gérer la communication des données entre deux environnements hétérogènes qui sont le client Android et le serveur.

Enfin, l'application que nous avons développée pourrait être enrichie par des fonctionnalités avancées telles que l'intégration d'un serveur FTP générateur d'applications, l'utilisation d'un protocole de communication plus sécurisé que le HTTP comme le HTTPS, mais aussi l'ajout d'autres services et mettre les applications sur le serveur web et ainsi encourager d'autres étudiants a développer toutes sortes d'applications ce qui agrandirait la portée de l'application.

Nous pouvons aussi, la rendre compatible avec plusieurs plateformes mobile, en la développant avec l'outil Adobe Flash qui génère des programmes d'extension SWF et qui est la tendance actuellement dans le développement des applications embarquées pour mobile.

Bibliographie

[1] http://fr.wikipedia.org/wiki/Android#R.C3.A9partition_des_versions
(Site consulté le 18 mai 2015)

[2] « Programmation des applications mobiles avec Android » cours de Olivier Le Goaer

[3] « Développement sous Android » cours de Jean Francois Lelande

[4] « Ingénierie Dirigée par les Modèles (IDM) – Etat de l'art » Benoit Combemale

[5] Portail des développeurs Android .
http://developer.android.com/

[6] http://laine.developpez.com/tutoriels/alm/mda/generalites-approche-mda/
(Site consulté le 10 mai 2015)

[7] « L'architecture dirigée par les modèles (MDA)» F.-Y. Villemin, CNAM

[8] http://android-pour-les-nuls.fr/tutoriaux/developpement/tuto-developpement-application-android-debutant (Site consulté le 04 mai 2015)

Liste des figures

Liste des abréviations

SMS Short Message Service

EDI Electronic Data Interchange

ADB Android Debug Bridge SDK Software Development Kit

NDK Native Developpement Kit

GPS Global Positioning System

XML eXtensibleMarkupLanguage

PNG Portable Network Graphics

OS operating system

MOF Meta-Object Facility

DSLM Domain Specific Modelling

Résumé

Dans le cadre de notre projet de fin d'études, nous étions menées à explorer ce nouveau système d'exploitation pour mobiles et à faire une application de messagerie électronique, téléphonique et geolocalisation. Notre objectif principal, était de développer une application Android permettant de générer automatiquement une autre application Android.

Le travail était établi en deux phases distinctes ; la première consistait à développer une application mobile « Cliente » qui permet de générer le modèle d'une futur application Mobile. La deuxième phase, consistait, et à partir du modèle généré dans la première phase, à développer une application « Serveur » qui permet à son tour de créer une nouvelle application Android.

Abstract

As part of our graduation project , we were conducted to explore this new operating system for mobile and make a mail application , phone and geolocation . Our main objective was to develop an Android application to automatically generate another Android app.

The work was developed in two phases ; the first was to develop a mobile application " Client " to generate the model of a future Mobile application . The second phase consisted, and from the model generated in the first phase, to develop a "Server" application that in turn create a new Android application.ى

ملخص

في اطارمشروع نهاية الدراسة كان علينا البحث و توسيع النضام التشغيل الجديد للهواتف النقالة والرسائل الالكترونية و الهاتفية, و كدالك (جي بي س) تحديد الموقع الجغرافي.

أول و أهم هدفنا كان تطوير تطبيقة أندرويد التي تسمح لنا بتوليد تاقائي لتطبيقة أندرويد أخرى

التطوير كان مرحلتين متفرقتين اللأولى تحث على تطوير تطبيقة موبايل من نوع زابون التي تسمح باعداد نوع التطبيقة الجديدة

والثانية تحث على, من خلال النوع الدي تحصلنا عليه في المرحلة الأولى تطور تطبيقة من نوع خادم التي تسمح بدورها بصنع تطبيقة أندرويد جديدة .