UNIVERSITE DE LARBI TEBESSI, TEBESSA

FACULTÉ DES SCIENCES EXACTES ET SCIENCES DE LA NATURE ET DE LA VIE

DÉPARTEMENT DES MATHÉMATIQUES ET INFORMATIQUE

LABORATOIRE DES MATHÉMATIQUES, INFORMATIQUE ET SYSTÈMES (LAMIS)

Thèse

Présentée en vue de l'obtention du diplôme de
Doctorat troisième cycle LMD

Titre de la thèse:

Contribution à L'évolution des Architectures Logicielles des Systèmes Intensifs

Discipline: Informatique

Spécialité : Systèmes d'information coopératifs

Soutenue publiquement par

Mlle Manel Gherari

Devant le jury ci-dessous :

Président :	Mr Hakim Bendjenna	MCA	Université de Tebessa
Directeur de thèse :	Mr Abdelkrim Amirat	Professeur	Université de Souk-Ahras
Co-directeur de thèse :	Mr Ridda Laouar	Professeur	Université de Tebessa
Examinateur :	Mr Makhlouf Dardour	MCA	Université de Tebessa
Examinateur :	Mr El Kamel Merah	MCA	Université de Khanchela
Invité :	Mr Mourad Oussalah	Professeur	Université de Nantes

REMERCIEMENTS

C'est avec un grand plaisir que j'apporte ce témoignage écrit de ma reconnaissance à tous ceux qui m'ont gratifié de leurs soutiens et de leurs confiances tout au long de ce travail.

Je tiens à remercier mon directeur de thèse Monsieur Amirat Abdelkrim, professeur à l'Université Souk-Ahras, de m'avoir accueilli dans son équipe et d'avoir accepté de diriger ce travail. Sa rigueur scientifique, sa disponibilité et ses qualités humaines m'ont profondément touchée.

Mes remerciements s'adressent également à Monsieur Ridda Laouar, professeur à l'université de Tebessa, pour le privilège qu'il m'a fait en acceptant de diriger ce travail. Je remercie également mon co-encadrant Monsieur Mourad Oussalah, professeur à l'Université Nantes pour avoir accepté de diriger ce travail. Son soutien, sa clairvoyance et ses compétences m'ont été d'une aide inestimable.

Je tiens à remercier sincèrement le président de jury docteur Hakim Bendjenna qui m'a fait le grand honneur d'évaluer ce travail, les membres du jury, docteur Merah EL-Kamel et docteur Darddour Makhlouf, pour m'avoir fait l'honneur de participer à mon jury de soutenance en tant qu'examinateurs.

Mes remerciements les plus chaleureux vont à tous mes camarades au LAMIS pour leurs encouragements et pour l'ambiance agréable tout au long de cette période et en particulier à Mlle Afrah Djeddar et Mlle Benatia Imene pour leurs présences dans les moments difficiles et grâce à qui j'ai passé d'excellents moments.

Je ne peux clore cette page de remerciements sans les adresser à: mes parents, qui durant ma scolarité m'ont toujours donné la possibilité de faire ce que je voulais et qui a toujours cru en moi.

Gherari Manel

RÉSUMÉ

Le Mobile Cloud Computing (MCC) a été introduit comme une solution optimale pour surpasser les limites des appareils mobiles. Le MCC déplace les appareils mobiles à un niveau entièrement nouveau où ces derniers comptent sur le Cloud pour stocker et traiter les données mobiles.

Les applications mobiles à base de Cloud englobent des fonctionnalités architecturales plus compliquées et plus riches que celles des applications mobiles traditionnelles, elles sont considérées comme la prochaine génération d'applications mobiles. Ainsi, de nouveaux défis surgissent dans leurs développements, comme les problèmes de communication avec des Cloud hybrides, la configuration dynamique, exploiter le concept de la mobilité et des informations contextuelles. Étant différente des applications traditionnelles, ce nouveau type d'application requiert de nouvelles méthodes, techniques et outils de développement, pour leurs modélisations et simulations. Face à ce manque dans les MCC, nous proposons dans cette thèse un Framework fondé sur le concept de Contexte-Awareness qui permet :

- La description de l'architecture des applications mobiles à base de Cloud avec Mobile Cloud Architecture Description Language (MC-ADL).
- La simulation de comportement des applications mobiles à base de Cloud dans un environnement Cloud simulé à l'aide de Mobile Cloud Simulation Toolkit (MC-SIM).
- Le provisionnement des services à travers un middleware sensibles au contexte: Smart Cloud Gate (SCG), pour assurer une configuration contextuelle lors de l'exécution des applications et aussi garantir une évolution contextuelle de leurs architectures.

Mots Clés: Mobile Cloud Computing; modélisation des applications mobiles; simulation architecture des applications mobiles; Context Awareness; Mobile Cloud Middleware.

ABSTRACT

Mobile Cloud Computing (MCC) has been introduced as an optimal solution to the constrained mobile devices. MCC takes mobile devices to a wholly new level where the latter rely on the cloud to store and process data.

Since mobile cloud applications encompass more complicated and rich architectural features, they are considered as the next generation of mobile applications. Thereby new challenges arise in their development phase, such as addressing issues of communication with the hybrid cloud, dynamic configuration, exploiting the concept of mobility and contextual information.

Being different from traditional applications, this new kind of applications requires new methods, techniques and tools of development, modeling and simulation. Addressing this lack in MCC, we propose in this thesis a context-aware based framework that allows:

- The description of mobile cloud applications architecture with the Mobile Cloud Architecture Description Language (MC-ADL).
- The simulation of mobile cloud application behavior in a simulated cloud environment using mobile cloud simulation toolkit (MC-SIM).
- The provision of a context-aware middleware: Smart Cloud Gate (SCG) to ensure contextual configuration of the mobile cloud application at runtime level and contextual evolution of its architecture.

Keywords: Mobile Cloud Computing; MCC Modeling, MCC simulation; Context Awareness; Mobile Cloud applications Architecture; Mobile Cloud Middleware.

ملخص

لقد تم تقديم Mobile Cloud Computing في مجال الاعلام الآلي كحل أنسب لتجاوز الحدود التي تفرضها الأجهزة النقالة.

MCC يقوم بنقلة نوعية لتكنولوجيا الهواتف النقالة أين هذه الأخيرة تتحرر من قيودها المفروضة بسبب الآلة و تقوم بالاعتماد الكلي على Cloud Computing من اجل تخزين و معالجة المعلومات.

بما أن تطبيقات الهاتف النقال التي تعتمد على Cloud Computing تتميز بعدة خصائص معقدة فيجب اعتبارها الجيل الجديد في تكنولوجيا الهاتف النقال لذلك فان عدة تحديات سوف تطرح مع كيفية تصميم وتصنيع هذا النوع الجديد من التطبيقات.

من بين التحديات المطروحة حاليا: كيفية التعامل مع Cloud Hybrid و كيفيةتطويرها ديناميكيا ,كذلك كيفية استغلال الحركية و المعلومات المرتبطة بالجهاز و محيطه.

ان اختلاف الجيل الجديد من التطبيقات عن الجيل القديم يكمن في تطلب التطبيقات الجديدة لأساليب و أدوات و تقنيات جديدة لتطويرها و تقديمها و محاكاتها. لمعالجة هذا النقص نقترح في هذه المذكرة نظام شبه متكامل مبني على استدراك سياق الهاتف.

النظام مصمم كما يلي:

1- تصميم تطبيقات الهواتف الذكية المبنية على CC باستعمال Mobile Cloud ADL .
2- محاكاة عمل هذه التطبيقات باستعمال نظام محاكاة MC-SIMالذي يعتبر ملحق لنظام المحاكاة . Cloud-SIM
3- Smart Cloud Gate التي تعتبر وسيط بين التطبيقات و Computing Cloud. هذه الأخيرة تستغل سياق عمل الهاتف من أجل ضمان تحديث و تطوير التطبيقات بشكل فعال كذلك مجاراة التطوير الذي يحدث على مستوى تصميمها.

TABLE DES MATIÈRES

Chapitre 2: Introduction au paradigme Mobile Cloud Computing et les Applications Mobiles à Base de Cloud

Chapitre 3: Modélisation et ingénierie dirigée par les modèles

Chapitre 4: Simulation Informatique d'un environnement Cloud

Chapitre 5: Conception et Réalisation

LISTE DES FIGURES

LISTES DES TABLES

INTRODUCTION GÉNÉRALE

1 Cadre de la thèse

Ce mémoire s'inscrit dans le cadre de l'étude des applications mobiles à base de Cloud, où nous nous intéressons plus précisément à leurs modélisations, la simulation de leurs comportements dans un environnement Cloud, et à comment introduire le concept de contexte dans leurs développements d'une façon avantageuse.

Les appareils mobiles ont envahi nos vies comme autant une nécessite et non plus un moyen de luxe. À ce rythme-là, et comme prédits par Walsh (Walsh 2010), les Smartphones dépasseront les Pcs, comme les moyens les plus utilisés pour accéder au World Wide Web. Ainsi, les appareils mobiles deviendront plus importants et seront impliqués dans presque tous les aspects de notre quotidien. De nos jours, les applications mobiles sont de plus en plus ubiquitaires et fournissent des services encore plus riches pour satisfaire les exigences toujours changeantes de l'utilisateur. Par conséquent, les limites des appareils mobiles (ex. Bande passante, la capacité de stockage, faible autonomie de la batterie et la fiabilité, etc.) sont un obstacle qui doit être surpassé. Ces problèmes sont résolus par le Cloud Computing qui peut être défini comme Utility Computing, où les applications sont fournies via l'internet en tant que services à l'aide d'une métrique de paiement.

Le Mobile Cloud Computing (MCC) est le résultat de la synergie entre le Cloud Computing et le Mobile Computing. Cette nouvelle technologie est considérée comme un grand saut vers un avenir prometteur pour le Mobile Computing, puisqu'elle vise à amener les avantages du Cloud Computing sur les appareils mobiles afin d'améliorer leurs performances. Le Cloud Computing a élevé le Mobile Computing à un niveau supérieur, où les calculs et le stockage sont externalisés dans le Cloud au moment de l'exécution à l'aide de technique de déchargement (Kumar and Lu 2010), et la délégation (Flores and Srirama 2013).

- Le Cloud Computing est un paradigme, encore à ses premiers balbutiements. Ainsi lors de son adaptation par le Mobile Computing, plusieurs questions sans réponse émergent, dont certaines portes sur :
- Comment diminuer les efforts et la complexité du développement d'une application mobile qui exige l'accès à des Clouds hybrides (Flores and Srirama 2013) ?
- Comment gérer un multi-Cloud sans surcharger les opérations des ressources mobiles ?

Et enfin les questions que nous jugeons les plus importantes sont :

- Comment modéliser cette nouvelle génération d'applications mobiles à base de Cloud ?
- Comment étudier leurs comportements dans un environnement Cloud?

- Comment prendre avantage du Cloud et du contexte mobile pour produire des applications performantes ?

Les développeurs des applications mobiles à base de Cloud (MC-Apps) sont limités à l'aspect de la mobilité et doivent faire face à un environnement d'exécution qui est soumis à des modifications constantes. Ce dernier, fournis des informations contextuelles qui amélioreront le développement des MC-Apps si elles sont utilisées de façon intelligente. La nécessité d'exploiter le contexte réside dans le fait qu'il fournit d'importantes informations sur l'état récent d'une personne, de sa localisation, et d'appareils avoisinants. Le principal défi de ce genre d'informatique mobile est de tirer parti de l'évolution de l'environnement avec une nouvelle classe d'applications conscientes du contexte dans lequel ils sont exécutés.

2 Problématiques

Dans cette section, nous allons mettre l'accent sur quelques limites du domaine étudié (c.-à-d. MCC) et leur proposer des solutions que nous jugeons adéquates.

Afin de surpasser les limites posées par les appareils mobiles, répondre aux besoins toujours changeants de l'utilisateur et développer des applications mobiles qui offrent des fonctionnalités assez similaires à celle offerte par les PC, le Mobile Cloud Computing (MCC) surgit. En déplaçant le traitement et le stockage de donnée depuis les appareils mobiles au niveau de Cloud, les problèmes liés à la durée de vie de la batterie, la consommation d'énergie et la performance sont résolus. En outre, nous avons conclu aussi que plusieurs autres problèmes émergent avec l'utilisation des Clouds. Parmi les problèmes non traités dans le domaine de MCC, nous citons notamment:

1) Les applications mobiles à base de Cloud ne sont pas considérées comme autant des applications assez compliquées et sophistiquées pour avoir une représentation architecturale et être modélisées;
2) Plusieurs simulateurs pour étudier et analyser le comportement des applications dans un environnement Cloud sont proposés dans la littérature, mais malheureusement non un seul simulateur dédié aux applications mobiles à base de Cloud a été proposé;
3) Pour améliorer la performance des applications mobiles à base de Cloud, les informations contextuelles sont très importantes, et pourtant elles ne sont pas pleinement exploitées.

3 Contributions

En répondant aux questions posées précédemment, nous présentons dans cette thèse la conception et la mise en œuvre d'un Framework pour la modélisation et la simulation des applications mobiles à base de Cloud (Context-Aware Framework For Modeling and Simulation of Mobile Cloud Applications).

Le Framework proposé comporte trois domaines principaux que nous soutenons les plus bénéfiques pour un développement efficace des applications mobiles à base de Cloud :

3.1 Modélisation des applications mobiles

Nous nous intéressons dans ce domaine à la modélisation et à la description des architectures des applications mobiles à base de Cloud (MC-Apps). Afin de fournir une vue abstraite des MC-Apps, pour faciliter au développeur de raisonner sur l'application avant son développement, nous proposons un outil (Mobile Cloud Architecture Description Language), qui permet d'identifier et de décrire les éléments entrant dans la conception de l'architecture de MC-App, les relations entre eux, et les contraintes sur leurs compositions et comportements.

Toujours dans le niveau abstrait, nous attestons que les informations contextuelles seront plus bénéfiques si elles sont modélisées et impliquées directement avec l'architecture de l'application, pour assurer un raisonnement efficace sur eux durant la vie de MC-App. Par la proposition de notre Framework, nous visons à développer des MC-App intelligentes, en termes bien spécifiques, une application réflective qui prend contrôle de son comportement dans des conditions variantes (c.-à-d. changement de contexte). Alors, la concrétisation de la relation entre l'application et le contexte est faite sous forme d'un contrat, entre les deux. *MC-App's Contract* est un modèle graphique qui décrit le comportement de MC-App dans une situation donnée créée par un changement de contexte.

3.2 Simulation des applications mobiles

Être étroitement lié à un environnement hétérogène, distribué à grande échelle, de plus payant, pose des défis hyper compliqués à affronter par les MC-Apps. En outre, la phase de test et de validation de ces dernières dans un environnement gratuit et contrôlable est antérieure et inévitable avant un déploiement réel dans l'environnement Cloud. Par conséquent, notre Framework fournit *Mobile Cloud Simulation Toolkit* (MC-Sim) qui est une extension du simulateur connu CloudSim(Calheiros *et al.* 2011). Le simulateur proposé répond au manque des simulateurs existants pour analyser et étudier le comportement des applications mobiles dans un environnement Cloud.

3.3 Gestion de complexité de traitement avec le Cloud

L'entrelacement avec le Cloud pose des problèmes sérieusement compliqués à savoir :

- La communication avec des Cloud hybrides ;
- L'administration des applications multi-Cloud qui nécessitent l'utilisation de plusieurs services Cloud de différentes plateformes;
- Répondre aux besoins des utilisateurs en gardant le niveau de qualité de service;
- L'autoadaptation des applications en fonction de changement dans l'environnement d'exécution.

En abordant ces problèmes, nous proposons un middleware contextuel *Smart Cloud Gate* (SCG) qui offre des services de composition, d'adaptation, et de reconfiguration. En plus, SGC assure une exploitation avantageuse du Cloud et une gestion intelligente des informations contextuelles.

4 Organisation du travail

En plus d'une introduction, ce mémoire est organisé en deux parties. La première partie intitulée « *Contexte de Travail* », présente les trois domaines sur lesquels focalise ce manuscrit. La deuxième partie intitulée « *Context-aware Framework for Modeling and Simulating Mobile Cloud Application* », décrit notre proposition et la validation de notre approche en termes d'expérimentations

1er Partie- Contexte de travail : Cette partie, expose les trois domaines étudiés. Elle comporte trois chapitres.

Chapitre 1 : Introduction au Mobile Computing : Approches et concepts de développement des applications mobiles

Ce chapitre introduit l'environnement mobile en présentant quatre sections principales. La première section porte sur la présentation des différents types des applications mobiles et mit en lumière leurs importances en fonction d'études statistiques. Dans la deuxième section, nous étudions les architectures des différents systèmes d'exploitation des appareils mobiles et nous présentons aussi les avantages et les inconvénients de chacun. Nous abordons dans la troisième section, les technologies utilisées pour développer les applications mobiles multiplateformes, ainsi une classification de ces derniers dans un tableau comparatif. Au niveau de la quatrième section, nous donnons une définition de la notion de « Contexte » mobile et nous présentons quelques exemples des applications mobiles conscientes au contexte.

L'objectif de ce chapitre est, d'identifier les concepts fondamentaux qui entrent dans le développement des applications mobiles, et les manques qui résident au niveau des approches proposées pour le processus de développement.

Chapitre 2 : Introduction au paradigme Mobile Cloud Computing et les Applications mobiles à base de Cloud

Ce chapitre pose les définitions données sur le paradigme « Mobile Cloud Computing », et comporte trois sections fondamentales. Dans la première section, nous abordons les définitions de Mobile Cloud Computing (MCC) et les détails de l'architecture proposée par Dinh et *al.* (Dinh *et al.* 2013) pour ce paradigme. La deuxième section présente les avantages apportés par le MCC. Nous introduisons dans la troisième section quelques exemples des applications mobiles à base de Cloud.

L'objectif principal de ce chapitre est de donner un aperçu sûr comment le MCC améliore la performance et le développement des applications mobiles et aussi mettre l'accent sur le fait que les solutions proposées dans la littérature dans le domaine du MCC, ne se positionnent pas sur l'axe architectural, mais plus tôt sur l'axe des middlewares.

Chapitre 3 : Modélisation et ingénierie dirigée par les modèles

Ce chapitre introduit la terminologie associée à l'ingénierie dirigée par les modèles et il est organisé en trois sections. La première section définit l'ingénierie dirige par les modèles (IDM) et la motivation derrière sa naissance. Aussi dans cette section, nous étudions les notions liées au Model Driven Architecture (MDA) qui représente un cas spécial d'IDM. Toujours avec la terminologie, dans la deuxième section nous présentons les définitions d'un modèle, méta modèle, la démarche de transformation de modèles dans l'IDM. Dans la troisième section, nous avons établi un lien entre la modélisation, l'ingénierie dirigée par les modèles et le développement des applications mobiles traditionnelles et celles à base de Cloud, nous discutons aussi les avantage que peut apporter l IDM au développement des applications mobiles et comment elle peut répondre au manque de méthode systématique de développement mobile.

Chapitre 4 : Simulation informatique d'un environnement Cloud

La simulation est un moyen efficace pour tester et étudier le comportement des systèmes dans un environnement ubiquitaire. Un des objectifs de ce travail est de fournir un moyen pour évaluer les performances des applications mobiles dans un environnement Cloud simulé. Avant d'entamer cette phase, nous allons étudier les simulateurs existants dans la littérature.

Ce chapitre, porte sur un état d'art sur les outils les plus connus dans la littérature pour simuler les systèmes informatiques dans un environnement Cloud. Premièrement, nous présentons chaque outil étudié en termes d'objectifs et d'architecture. Deuxièmement, nous établions une classification des outils par rapport aux détails de mise en œuvre (ex. Langage de programmation, plateforme d'implémentation). Il est à noter que, l'objectif principal de ce chapitre est de mettre en lumière le fait qu'il manque de simulateurs dans le domaine du Mobile Cloud Computing. Aussi, de choisir le simulateur le plus adéquat comme sujet d'extension en se basant sur une étude approfondie sur cet axe de recherche.

2eme Partie- Context-Aware Framework for Modeling and Simulating Mobile Cloud Applications: En se concentrant sur son unique chapitre, la deuxième partie de ce manuscrit se focalise sur le Framework que nous proposons pour modéliser et simuler les applications mobiles à base de Cloud.

Chapitre 5: Conception et réalisation

Ce chapitre, présente *Context-Aware Framework for Modeling and Simulating Mobile Cloud application*. Il est organisé en quatre sections principales.

La première section aborde la problématique que nous avons identifiée dans le développement des applications mobiles (c.-à-d. manque d'outils et de méthodes de modélisation). La deuxième section présente nos apports et constructions en termes de solutions aux problématiques visées. Une vue générale du Framework proposé est aussi donne au niveau de la deuxième section. Nous détaillons dans la troisième section les outils constructifs du Framework proposé (MC-ADL, SCG, MC-SIM), en présentant les motivations derrière la proposition de chaque outil, son architecture interne, et les détails

de sa mise en œuvre. Enfin dans la quatrième section, nous évaluons notre approche par le bai d'une étude de cas comparatif.

En guise de conclusion, nous faisons le bilan de ce travail, pour souligner très précisément nos principaux apports et contributions apportées par cette thèse. Ensuite, nous procédons à une critique de ce travail et proposons alors des extensions possibles.

Chapitre 1

Introduction au Mobile Computing : approches et concepts de développement des applications mobiles

1.1 Introduction

À propos de l'émergence des appareils mobiles, deux discours connexes existent dans la communauté technologique. Selon le premier discours, les Smartphones et les tablettes tendent de plus en plus à remplacer les ordinateurs personnels en tant qu'appareil le plus utilisé pour accéder à Internet. Nous entrerions donc dans l'ère de postordinateur personnel ou dans une ère dominée par la technologie mobile.

Le second discours porte sur le comportement des consommateurs dans un monde multiplateforme. En admettant que la technologie mobile domine actuellement le marché. Nous évoluons désormais dans un univers multiplateforme dans lequel les utilisateurs utilisent à la fois l'ordinateur personnel, téléphone intelligent et la tablette, en passant facilement de l'un à l'autre.

Il est à noter que les deux discours sont soutenus par des recherches. Peu importe lequel des deux discours correspond le mieux à la réalité, les appareils mobiles sont en fait au cœur d'une nouvelle utilisation d'Internet, plus personnelle et localisée, qui bouleverse profondément ce qu'il conviendrait d'appeler l'ancien ordre numérique.

Si l'importance du mobile est aujourd'hui reconnue dans la plupart des entreprises, qu'en est-il des applications mobiles ? Où en est l'usage et que représente ce marché ?

Revenons un peu en arrière, quelque part en 2008, Steve Jobs annonçait lors de la conférence keynote d'Apple sa principale nouveauté, App Store, le magasin en ligne des applications tierces pour iPhone et iPod touch. La boutique renfermait seulement 500 applications à son ouverture (Apple 2008). De son côté, l'Android Market (qui deviendra Google Play en 2012) rassemblait à ses débuts près de 2 300 applications en 2009 (PC-world 2010). Aujourd'hui, plus de 1,3 million sont présent dans le Google Play Store, 1,2 million dans l'Apple App Store, 300 000 dans le Windows Phone Store et ces nombres des applications disponibles ne cessent d'augmenter d'année en année (Statista 2016).

Dans ce chapitre, nous présentons les applications mobiles et leurs différents types, les systèmes d'exploitation des appareils mobiles et leurs architectures, ainsi que les avantages et les inconvénients de chaque système. Enfin, nous abordons les technologies multiplateformes dédiées au développement des applications mobiles.

1.2 Introduction aux applications mobiles

1.2.1 Définition d'une application mobile

Une application mobile ou « Apps » n'est apparu qu'avec l'introduction d'IPhone en 2007. Les applications mobiles sont des logiciels applicatifs développés pour être installés sur un appareil mobile, tels qu'un assistant personnel, un téléphone portable ou un Smartphone. Une telle application peut être installée sur l'appareil ou peut être téléchargée par l'utilisateur en ligne à partir de Google Play ou l'App Store (Minelli and Lanza 2013).

1.2.2 Caractéristiques des applications mobiles

Il est à noter que, la conception et l'implémentation des applications mobiles sont fortement liées à la caractéristique de l'appareil dont lequel ces applications vont être installés, mais reste toujours quelques caractéristiques en commun entre les applications mobiles malgré la différence en terme d'appareil et de système d'exploitation.

- Temps de lancement: L'application doit se lancer en moins de 5-6 seconds.
- Haute performance: Garantir constamment une haute performance et une rapide réactivité afin de répondre aux exigences de l'utilisateur qui cherche la fiabilité dans l'application utilisée.
- Pas d'étapes intermédiaires afin d'atteindre le but d'exécution de l'application.
- La consommation d'énergie: L'optimisation du processus de l'application pour garantir une exploitation efficace de l'énergie.

1.2.3 Les applications mobiles en chiffres

Depuis le lancement des boutiques des applications mobiles, ces dernières ne cessent pas de s'augmenter. Dans cette section nous présentons quelques statistiques récentes à propos de téléchargement et d'utilisation des applications mobiles, pour mettre l'accent sur leurs importances et leurs popularités.

Le secteur européen des applications mobiles représentera d'ici à 2018, 63 milliards d'euros et près de 5 millions d'emplois selon un rapport de l'UE. Les revenus vont être multipliés par 3,6 et les emploient par 2,7 en seulement 4 ans.

Android et iOS totalisent 84% du marché des systèmes d'exploitation pour smartphones en janvier 2015, suivi de Windows avec 9%. Android (63%) est en augmentation de 7%, iOS (21%) perd 1% sur 1 an alors que Windows gagne 2%.

76%, c'est l'augmentation moyenne de l'usage des applications mobiles en 2014 d'après *Flurry Analytics* (voir figure 1.1) (Apps 2016).

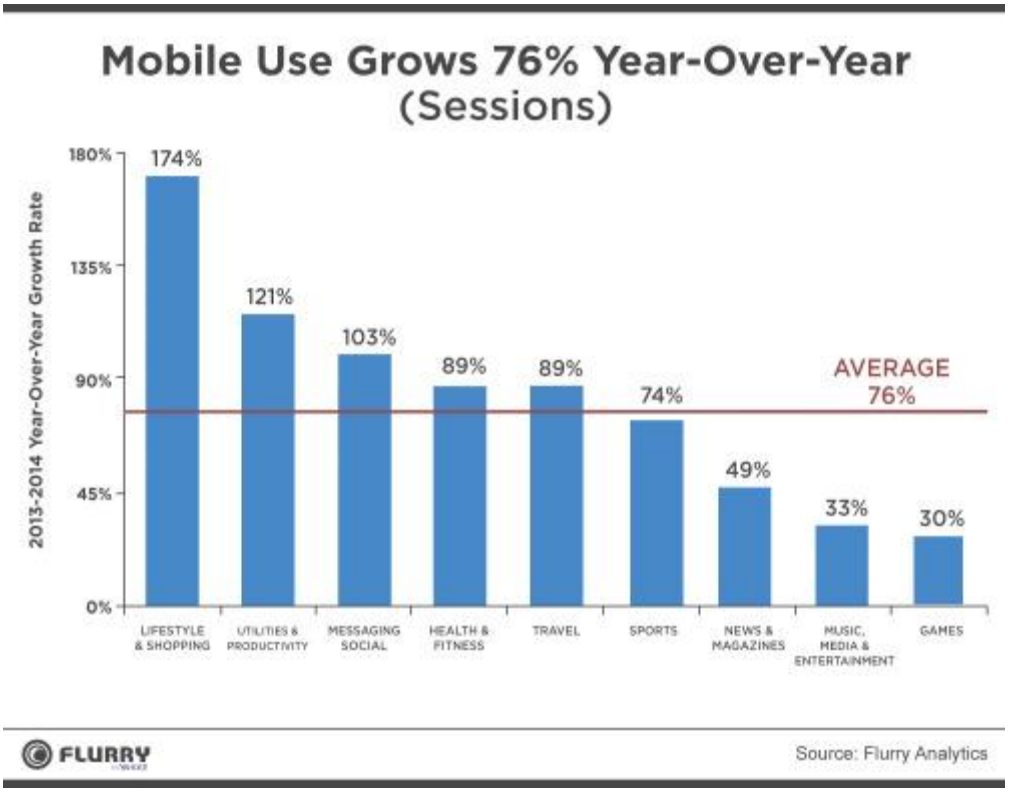


Figure 1.1 L'utilisation des appareils mobiles.

Les applications de la catégorie « life style & shopping » comme montré dans la figure 1.2 ont par exemple augmenté de 174% leurs usages (220% pour la catégorie Android « shopping » seul). 86% du temps passé sur le Web Mobile est généré par les applications, c'est **6%** de plus qu'en 2013. Les utilisateurs Android et iOS passent 32% de leurs temps sur des applications de jeux, 17% de leurs temps sur Facebook et 9,5% sur de la messagerie sociale.

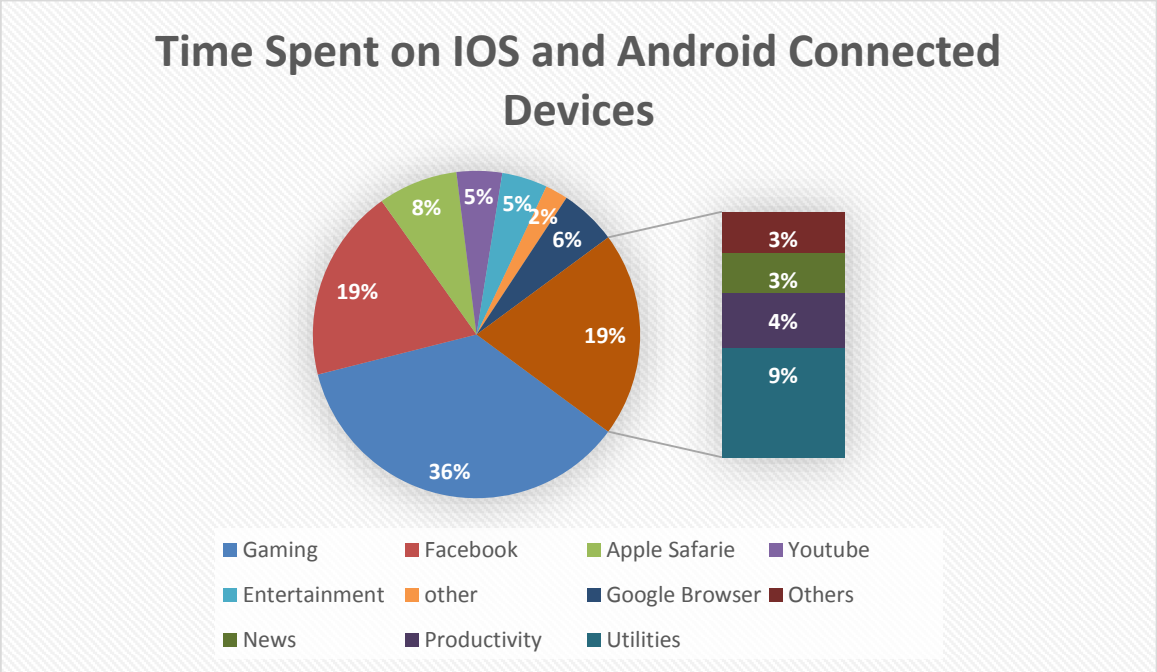


Figure 1.2 Le temps passé sur les appareils mobiles (Apps 2016).

Sur 35 applications mobiles installées en moyenne par Smartphones, 11 restent utilisées toutes les semaines, mais ce sont désormais 12 applications (contre 10 fin 2012) qui ne sont jamais utilisées. Début 2014, 15% des mobinautes (22% fin 2012) installent encore au moins une nouvelle application toutes les semaines. 41% désinstallent des applications au moins une fois par mois contre 44% fin 2012.

Ces chiffres montrent que le marché représente un très gros potentiel pour les entreprises et que l'usage des mobinautes atteint aujourd'hui une certaine maturité. L'enjeu est désormais de retenir les utilisateurs dans un marché des applications de plus en plus vaste.

1.3 Types d'applications mobiles

La création d'une application mobile amène beaucoup de questions concernant le développement, fonctionnalité et l'ergonomie. Alors avant de commencer de développer une application mobile, il est indispensable de bien choisir quel type d'application à développer et bien argumenter notre choix. Dans les sous-sections suivantes, nous présentons les différents types d'application mobile avec une étude comparative.

1.3.1 Applications natives

Les applications natives sont des programmes développés pour exécuter certaines fonctionnalités et être déployables dans des plateformes ou appareils mobiles particuliers. Ces derniers permettent aux applications de prendre avantage de leurs caractéristiques, leurs applications préinstallées par défaut (ex. Calendrier et contact), et aussi les technologies qu'elles fournissent comme la caméra et le GPS.

Les applications natives peuvent être téléchargées depuis des boutiques en ligne publiques ou privées, ensuite être installées sur l'appareil mobile ou bien elles peuvent exister par défaut avec l'appareil mobile (ex. Contact, Messagerie et Calendrier, etc.).

1.3.1.1 Avantages

- Une meilleure performance par ce qu'elles utilisent les logiciels et le matériel valable dans l'appareil mobile
- Comme ces applications sont déjà installées dans l'appareil et utilisent les données déjà existantes, elles peuvent être exécutées hors ligne sans besoin d'Internet
- Chaque application est distinguée par son logo pour attirer l'attention de l'utilisateur.

1.3.1.2 Inconvénients

- L'inconvénient majeur de ce type d'application, c'est qu'elles sont fortement liées avec l'appareil dans lequel elles sont installées, ce qu'engendra l'impossibilité d'évoluer ou exploiter des nouvelles technologies.

- Rendre les applications natives déployables dans différents types d'appareils/ plateformes (c.-à-d. réécrire le code dans différents langages) est une tache consommatrice en termes de temps.
- L'exclusivité des applications natives pour un type d'appareil mobile bien particulier diminuera le nombre des utilisateurs et le gain de leurs ventes

1.3.2 Applications mobiles web

Les applications mobiles web sont une version réduite des applications web dédiées pour les appareils mobiles. L'accès à ce genre d'application est fait par un navigateur web. Par conséquent, ces applications sont indépendantes des plateformes et caractéristiques des appareils mobiles. Évidemment, ces applications exigent une connexion internet pour leurs utilisations.

Lors de développement des applications mobiles web, il est essentiel de:

- Définir le texte et les images statiques de l'application en utilisant HTML5
- Définir le style et la présentation de l'application en utilisant CSS
- Définir l'interaction et l'animation en utilisant le JavaScript

Une meilleure utilisation de ce genre d'application est due au fait de la possibilité d'une réorganisation de leurs contenus en fonction des caractéristiques de l'appareil mobile.

1.3.2.1 Avantages

- L'avantage le plus important des applications mobiles web est: la capacité de déploiement dans multiples plateformes indépendamment du type d'appareil mobile
- Moins cher, facile et rapide à développer
- Accès facile en utilisant un simple URL sans le besoin d'installation ou le téléchargement des compléments.

1.3.2.2 Inconvénients

- Les navigateurs traditionnels sont plus performants que les navigateurs des appareils mobiles.
- Les applications mobiles web ne peuvent pas exploiter les fonctionnalités offertes par les logiciels et les hardwares des appareils mobiles.
- La performance des applications mobiles web dépend de la vitesse et l'état de la connexion Internet.

1.3.3 Applications mobiles hybrides

Les applications mobiles hybrides sont la combinaison entre les applications natives et les applications mobiles web. La synergie entre ces deux types d'application résulte une réduction en temps, effort de développement, prix et la maintenance. Ces applications sont téléchargeables via des boutiques en ligne, installées sur l'appareil et exécutées

depuis une simple icône comme les applications natives. Les applications mobiles hybrides sont créées pour être exécutées sur plusieurs plateformes.

1.3.4 Applications natives vs. Applications mobiles web

Le tableau ci-dessous énumère les principales considérations techniques et commerciales qui montrent les différences entre une application native et une application mobile web (rabidvaluesolution 2016).

	Native Apps	Web Apps
Development Cost	Due to the diversity of targeted devices and operating systems it is usually more expensive.	Since it is a cross-platform app, it is considered usually less expensive.
Development Resources	May require multiple developers due to the differences in the needs of each operating system.	Build once and deployed everywhere since it is cross-platform app.
Ease of Development	Requires distinct Software Development Kit (SDK) and programming languages for each device.	Developed using the traditional web tools.
Update and Maintenance	May need more development resources based on the number of platforms and devices.	Easy to update and getting the new version just by refreshing the browser.
Audience Reach	Only users with specific devices.	Everyone with a mobile browser.
Ease and Speed of Implementation	Must be downloaded and installed before use.	Published as a regular website with no need to download.
Installation	Downloaded and installed from a website or a marketplace.	No installation required.
Access to Hardware Functionalities	Can access almost all the functionalities of a device: camera, microphone...	Only GPS can be accessed so far for most of the apps.
Paid vs. Free	Easy to handle via app marketplace.	It is a bit difficult since the mobile browsers are for all the devices.
Search	Easy search in the marketplace.	Search for a normal website with the ability to redirect to the mobile version when the mobile device is detected.
Internet Connectivity	Can work offline.	It is rigorously required.

Table 1. 1 Native Apps vs. Web Apps (rabidvaluesolution 2016).

1.4 Les systèmes d'exploitation des appareils mobiles

Un système d'exploitation est un ensemble de programmes qui dirige l'utilisation des capacités d'un ordinateur par les logiciels applicatifs. Cette définition s'applique aussi aux Smartphones que nous retrouvons sur le marché, car ils possèdent les mêmes caractéristiques qu'un ordinateur à savoir une mémoire interne de stockage, un système d'exploitation et un processeur (Ekito 2012).

Le principal rôle d'un OS est de gérer la mémoire et les périphériques (clavier, écran…). Il détecte les actions de l'utilisateur et transmet l'information à la machine. Le monde du dispositif mobile est contrôlé par quatre systèmes d'exploitation : Apple IOS, Google Android, BlackBerry OS et Windows Phone. Chaque système d'exploitation possède des fonctions distinctives pour leurs utilisateurs. Android et IOS d'Apple se taillent la plus grande part de marché comme nous le montre le tableau ci-dessous (Table 1.2).

Le marché du Smartphone dans le monde entier a augmenté de 13,0 % sur un an en 2015 Q2, avec 341,5 millions d'expéditions, selon les données de l'International Data Corporation (IDC). Cette croissance est principalement attribuable à des gains enregistrés sur les marchés émergents tels qu'APEJ et MEA.

	Android	**iOS**	**Windows Phone**	**BlackBerry OS**	**Autres**
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Table 1. 2 Ventes mondiales d'appareils par système d'exploitation »

Ce trimestre, les expéditions ont été légèrement inférieures aux prévisions et IDC s'attend à voir un ralentissement notable dans le Smartphone d'expéditions en 2015 que la Chine se joint à l'Amérique du Nord et en Europe de l'Ouest dans un profil de croissance plus mature. Android domine le marché avec une part de 82,8% en 2015 Q2. Samsung a réaffirmé son rôle de leader mondial avec un accent renouvelé sur les Smartphones à moindre coût (Mobile 2016).

1.4.1 IOS

L'iOS, précédemment connu sous le nom OS d'iPhone, a été développé par Apple et peut être trouvé dans tous les produits Apple (iPhone, iPod Touch, iPad). Il est codé en langage Objective-C. La dernière version d'OS est l'iOS 9 qui vient d'être présenté par Apple (Apple 2016).

1.4.1.1 Architecture d'iOS

Comme la figure 1.3 le montre, nous remarquons qu'iOS comporte cinq couches d'abstraction : *Application, Core OS, Core Services, Media, Cocoa.* Le système d'exploitation occupe moins d'un demi-gigaoctet (Go) de la capacité mémoire totale de l'appareil. Nous expliquons dans ce que suivent les couches en détail.

a) Applications

La couche application regroupe les applications natives préinstallées et les applications tierces téléchargées depuis l'App Store. Citons par exemple phone, message, home…etc.

b) Cocoa Touch

Cette couche est un UI Framework responsable de la construction de logiciels à exécuter sur le système d'exploitation. Cocoa Touch contient principalement les classes implémentées en Objective-C, un langage orienté objet qui est compilé pour fonctionner à une vitesse incroyable, mais utilise un Runtime vraiment dynamique. Parce qu'Objective-C est un surensemble de C, il est facile de mélanger C et même C ++ dans les applications tactiles de cacao. Le Framework de cette couche fournit l'infrastructure des applications de base et de soutien pour les technologies clés (c.-à-d. Multitasking, entrée à base de touche, push notification et de nombreux services de système de haut niveau).

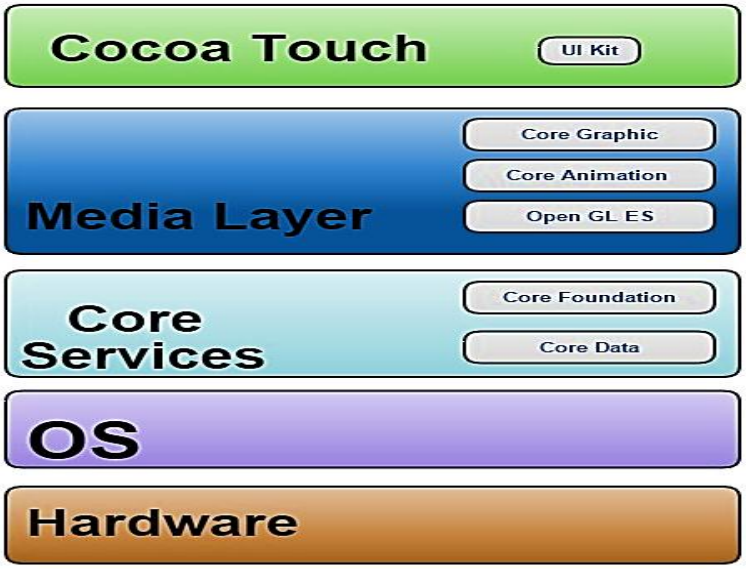


Figure 1. 3 Architecture d'IOS (Apple 2016).

c) Media

Couche responsable de la gestion multimédia comme l'audio et la vidéo, mais aussi les aspects graphiques comme le support d'OpenGL pour l'affichage 2D et 3D.

d) Core Services

Cette couche contient les services de base du système pour les applications. *Core fondation* and *Core Data* sont les services clés fournis par cette couche, pour définir les

types de base utilisés par toutes les applications. Aussi, elle regroupe les technologies qui supportent les fonctionnalités comme localisation, ICloud, Social media et Networking.

e) Core OS

Cette couche regroupe les fonctionnalités de bas niveau, sur lesquelles basent les fonctionnalités de haut niveau. Les services de cette couche ne sont pas utilisables directement par l'application, mais par les Frameworks utilisés par l'application. Principalement l'utilisation de cette couche est pour la gestion de la sécurité et la communication avec des hardwares externes à l'appareil mobile.

1.4.2 Android

Android est une plateforme logicielle et système d'exploitation développé par Google (2007). Android est basé sur le noyau Linux afin d'être exploité par une grande variété d'appareils mobiles.

La popularité d'Android (voir figure 1.4) et due au fait qu'il peut être trouvé sur une gamme d'appareils de différents fabricants notamment est, Samsung, Motorola et HTC, aussi Android accroit son avance technologique sur des mobiles largement moins chers.

Parmi les différentes versions d'Android, il y a KitKat, Jelly Bean, Honeycomb, Gingerbread et la dernière Lollipop 5.0 (Android 2016).

1.4.2.1 Architecture d'Android

Android est une pile applicative pour les appareils mobiles qui comprennent un système d'exploitation, middleware et des applications clés. Le SDK Android fournit les outils et les API nécessaires pour commencer à développer des applications sur la plateforme Android en utilisant le langage de programmation Java.

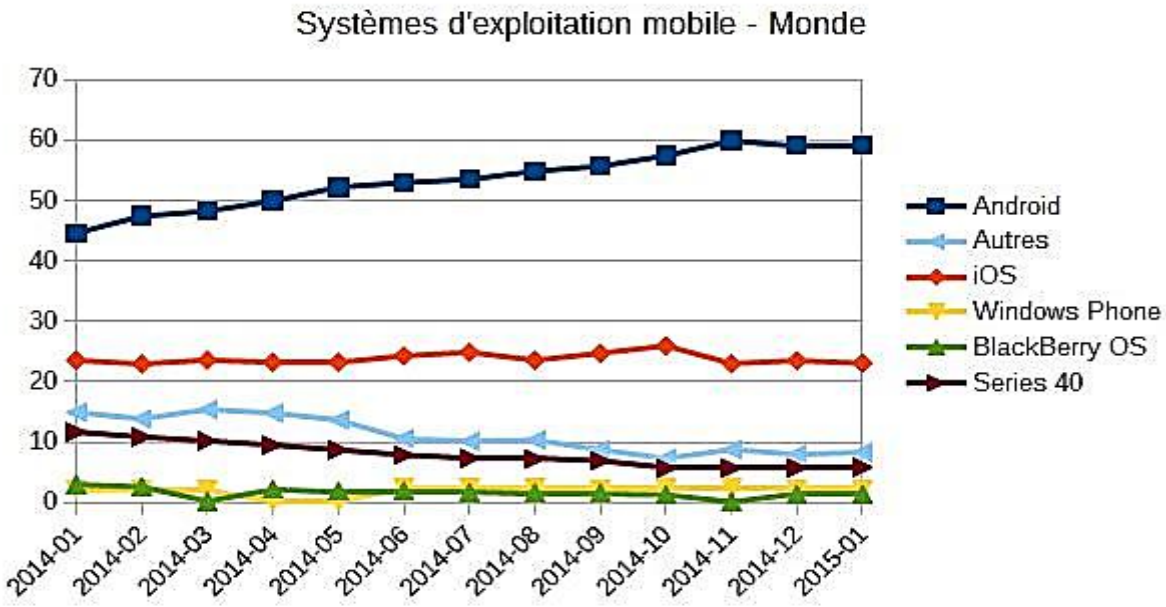


Figure 1.4 Systèmes d'exploitation mobile en monde (iwebyou 2015).

Le schéma suivant (Figure 1.5) illustre les principaux composants du système d'exploitation Android. Chaque section est décrite plus en détail ci-dessous.

a) Applications

Android est fourni avec un ensemble de programmes de base (également nommés applications natives) permettant d'accéder à des fonctionnalités comme les courriels, les SMS, le téléphone, le calendrier, les photos, les cartes géographiques, le Web, pour ne citer que quelques exemples. Ces applications sont développées à l'aide du langage de programmation Java. Pour l'utilisateur final, c'est la seule couche accessible et visible.

b) Le Framework (*Application Framework*)

En fournissant une plateforme de développement ouverte, Android offre aux développeurs la possibilité de créer des applications extrêmement riches et innovantes. Les développeurs sont libres de profiter du matériel périphérique, les informations de localisation d'accès, exécutez les services d'arrière-plan, définir des alarmes, ajouter des notifications de la barre d'état, et beaucoup, beaucoup plus.

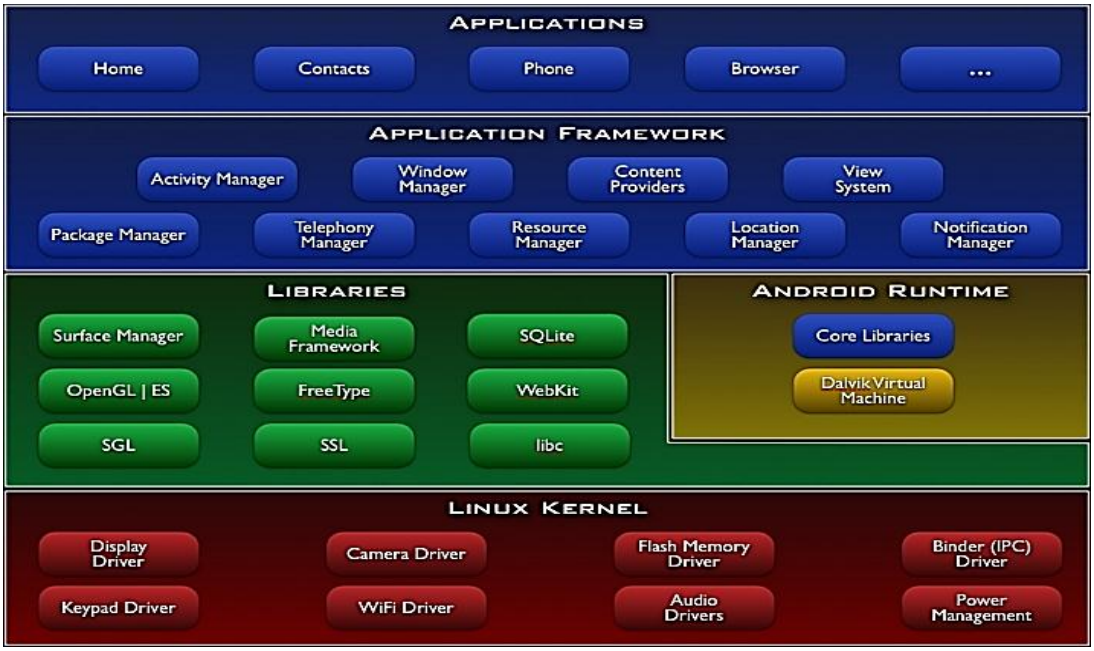


Figure 1.5 L'architecture d'Android (Android 2016).

c) Les bibliothèques (*Libraries*)

En interne, Android inclut un ensemble de bibliothèques C et C++ utilisées par de nombreux composants de la plateforme Android. Ces bibliothèques sont en réalité accessibles aux développeurs par l'intermédiaire du Framework Android. En effet, le Framework Android effectue, de façon interne, des appels à des fonctions C/C++ beaucoup plus rapides à exécuter que des méthodes Java standard. La technologie Java Native Interface (JNI) permet d'effectuer des échanges entre le code Java, le code C et C++. La liste ci-dessous énumère quelques-unes des bibliothèques disponibles dans Android :

- *Bibliothèque système C*: Implémentation (dérivée de BSD) de la bibliothèque standard C (libc), optimisée pour les systèmes Linux embarqués.
- *Bibliothèques multimédias*: Basées sur StageFright, elles permettent le support de nombreux formats audio et vidéo, tels que MPEG4, H.264, MP3, AAC, AMR, JPG et PNG.
- *SurfaceFlinger:* Permets l'accès au sous-système d'affichage.
- *LibWebCore:* Moteur de rendu de pages Internet basé sur Webkit. Cette bibliothèque est donc principalement utilisée dans le navigateur et dans les vues web embarquées (WebView).
- *Skia:* Moteur graphique 2D.
- *Bibliothèques 3D:* Implémentation basée sur OpenGL ES 1.0 API et plus récemment OpenGL ES 2.0.
- *FreeType:* Rendu des polices de caractères.
- *SQLit:* Base de données légère et puissante.

d) Moteur d'exécution Android (*Android Runtime*)

Android inclut un ensemble de bibliothèques qui fournit la plupart des fonctionnalités disponibles dans les bibliothèques de base du langage de programmation Java.

Chaque application Android s'exécute dans son propre processus, avec sa propre instance de machine virtuelle Dalvik. Dalvik VM est une implémentation de machine virtuelle ayant été conçue pour optimiser l'exécution multiple de machines virtuelles. Elle exécute du bytecode qui lui est dédié : le bytecode dex. (Format qui est optimisé pour une empreinte mémoire minimale).

Cette particularité d'Android en fait un système unique, loin des systèmes Linux traditionnels que beaucoup avaient pu rencontrer auparavant (Android 2016).

e) Noyau Linux (*Linux Kernel*)

Android repose sur un noyau Linux (version 2.6) qui gère les services du système, comme la sécurité, la gestion de la mémoire et des processus, la pile réseau et les pilotes. Il agit également comme une couche d'abstraction entre le matériel et la pile logicielle.

1.5 Les technologies multiplateformes

Dans cette section, nous allons définir la notion de multiplateforme en nous basant sur les logiciels, car les applications mobiles sont assimilables à ces derniers, ensuite nous allons tenter de relater l'intérêt du développement multiplateforme en toute généralité.

Un logiciel multiplateforme est un logiciel conçu pour fonctionner sur plusieurs plateformes. En anglais on parle souvent de « *cross-platforms software* ». Donc le développement multiplateforme peut être défini comme le procédé par lequel un développeur conçoit un logiciel pouvant être déployé sur différentes plateformes.

1.5.1 Les solutions dominantes « logiciels de développement »

1.5.1.1 PhoneGap

PhoneGap est une solution hybride. C'est une bibliothèque permettant d'accéder depuis le JavaScript d'une page web à de nombreuses ressources matérielles de l'appareil. PhoneGap ne gère pas l'interface utilisateur alors la présentation se fait à l'aide de CSS ou des langages spécialisés tels que Mobl (Desnos Decembre 2011).

Critiques

+ Une petite bibliothèque.

+ Des accès à de nombreuses ressources matérielles.

+La gratuité.

- Windows phone 7 n'est pas supporté pour le moment.

1.5.1.2 Titanium

Titanium est une plateforme JavaScript éditée par « *Appcelerator* », elle permet de créer des applications natives et offre une API supportant une grande quantité de ressources matérielles. *Appcelerator* fournit également un IDE basé sur Éclipse, « *Titanium Studio* », qui permet de compiler et tester son code à la volée. Cependant, Titanium souffre de deux inconvénients majeurs. Le premier est l'existence du connexion Internet ou cours d'utilisation du « *Titanium Studio* ». Le second inconvénient est le manque de la documentation (Desnos Decembre 2011).

1.5.1.3 Mobl

Mobl est un langage récent déclaratif et impératif, il permet de simplifier la construction d'une application. Le langage est compilé via un plug-in dans Éclipse ou en ligne de commande qui génère des fichiers HTML/CSS/JavaScript. Il faut noter cependant que l'aspect d'une application Mobl reste éloigné d'une application native.

Critiques

+ La rapidité du développement.

+ La simplicité du code.

- La documentation pauvre.

1.5.1.4 Corona

Corona permet de créer des applications très performantes en un minimum de code. Il permet d'avoir accès à une bonne partie des ressources matérielles. L'interface graphique est composée de deux couches, la première est native citons par exemple l'alerte et champ de saisie et la deuxième en OpenGL (listes et barres de navigation). L'inconvénient majeur de Corona c'est qu'il est dédié seulement pour le développement des jeux.

1.5.2 Tableau comparatif des solutions

Le tableau suivant compare les solutions entre elles. Il indique la disponibilité de ces dernières pour les principales plateformes du marché.

	Editeurs	Phone-Gap	Titanium	Mobl	Corona
IOS	Apple	+	+	+	+
Android	Google	+	+	+	+
BlackBerry Os	RIM	+	+	+	-
Symbian	Nokia	+	-	+	-
Windows Phone 7	Microsoft	-	-	-	-

Table 1. 3 Systèmes d'exploitation supportés par les solutions (Desnos Decembre 2011).

1.6 L'informatique mobile et le contexte

Nous avons, en tant qu'humains, une compréhension intuitive du contexte. Nous réagissons différemment en fonction d'éléments tels que : l'heure, la météo, l'expression du visage de l'interlocuteur, etc. Hoareau et Satoh soulignent dans leurs travaux (Hoareau and Satoh 2009) le fait que nous utilisons souvent, parois mêmes de manière inconsciente, des informations de contexte. Ils remarquent que pour y arriver, nous sommes dotés d'un système sensoriel complexe, constitué par des capteurs (les yeux, la peau, etc.) et une unité centrale qui traite les informations (le cerveau). Ainsi, en fonction des informations reçues nous adaptons notre comportement aux conditions externes (Popovici 2012).

Dans les deux dernières décennies, avec l'évolution de l'informatique mobile, la notion de contexte est également devenue très importante dans la recherche. Nos terminaux informatiques nous suivent partout et se doivent d'intégrer cette nouvelle notion, pour nous proposer des applications mobiles pertinentes.

1.6.1 Définition du contexte

Avant de passer à la définition des applications conscientes au contexte, il est indispensable de définir la notion du contexte. Des travaux très pertinents sont fournis en 1999-2001 par Dey (Abowd *et al.* 1999, Dey 2001). Les auteurs essayent d'améliorer la compréhension des notions *context* et *context-aware* en faisant une synthèse des travaux existants à l'époque, mais aussi en proposant leurs propres définitions (Chen and Kotz 2000) (Baldauf *et al.* 2007). D'après ces travaux :

«Le contexte est toute information qui peut être utilisée pour caractériser la situation d'une entité. Une entité est une personne, un lieu ou un objet qui est considéré pertinent pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application.»

Cette définition généralise les précédentes, qui se contentaient d'énumérer des éléments de contexte tel que: la position, l'identité des personnes et des objets voisins, les changements de voisins, la saison, la température, le temps comme dans (Brown *et al.* 1997) et (Nick *et al.* 1997). La définition donnée par Dey et Abowd permet de juger si une certaine information fait partie du contexte ou non. De plus, le contexte est considéré par rapport à chaque application (ou activité).

Dans le travail de Dey et Abowd (Abowd *et al.* 1999), les entités impliquées dans des interactions sont divisées en trois catégories : des lieux (bâtiment, pièce, etc.), des personnes (individus ou groupes) et des objets (objets physiques, ordinateurs). Une certaine information fait partie du contexte si elle influe sur une interaction entre deux entités.

Dans les travaux qui touchent au sujet du contexte, les auteurs ont des approches différentes en termes de modèle ou domaine d'application, comme nous allons le présenter dans la suite. Par contre, une grande partie des chercheurs s'appuient sur les définitions données par Dey et ses co-auteurs : (Gu *et al.* 2004), (Bolchini *et al.* 2007), (Baldauf *et al.* 2007), (Bettini *et al.* 2010), etc.

1.6.2 Définition de Context-Awareness

Le terme anglais *aware* signifie conscient. Une application peut être consciente de son contexte et réagir en conséquence. Les notions context-aware application et *context-aware computing* se traduisent respectivement par application sensible au contexte et informatique contextuelle. D'après (Abowd *et al.* 1999), ce serait communément reconnu dans la littérature que le premier travail de recherche en informatique sur le context-aware serait celui de Want et *al.* (Want *et al.* 1992), en 1992. Ce travail porte sur un système de localisation des employés dans les pièces d'un bâtiment, utilisant des badges actifs.

La première fois que l'on a parlé de *context-aware computing* est en 1994, dans le travail de Schilit et Theimer (Schilit and Theimer 1994). Les auteurs considèrent l'informatique contextuelle comme la capacité d'une application mobile à découvrir et à réagir à des changements dans l'environnement. D'autres mots qui décrivent l'informatique contextuelle incluse réactif (Cooperstock *et al.* 1995) ou réceptif (Elrod *et al.* 1993). Nous reprenons de nouveau la définition donnée par Dey et Abowd (Abowd et al. 1999):

« Un système est sensible au contexte s'il utilise le contexte pour fournir des informations et/ou des services pertinents pour l'utilisateur, où la pertinence dépend de la tâche de l'utilisateur. »

De nouveau, la définition est assez générale pour permettre d'inclure tous les aspects. La sensibilité au contexte peut ensuite être divisée en fonction de son utilisation. D'un côté, une application peut se concentrer sur la détection, la perception et l'interprétation des éléments de l'environnement de l'utilisateur, par exemple dans les travaux de Pascoe et *al*. (Pascoe 1998). D'un autre côté, les applications peuvent changer dynamiquement

leurs comportements, en fonction du contexte de l'utilisateur et de l'application par exemple dans (Brown *et al.* 1997) (Davies *et al.* 1998).

Les caractéristiques des applications sensibles au contexte seraient, d'après Dey (Dey 2001): présenter des informations et services à l'utilisateur ; exécuter automatiquement un service pour l'utilisateur ; étiqueter l'information de contexte pour une récupération ultérieure.

1.6.3 Les applications mobiles conscientes au contexte

Dans cette section, nous présentons quelques exemples d'applications mobiles conscientes au contexte.

1.6.3.1 Conscience basée sur le contexte de travail

Dans un environnement hospitalier avec de nombreuses activités d'importance variable, de l'urgence et du personnel avec de diverses compétences et responsabilités qui ont besoin d'une coordination, Bardram Hansen (Bardram and Hansen 2010) ont développé un système en milieu de travail basé sur le contexte pour atteindre efficacement cet objectif. Leur architecture est à la fois AwarePhone et AwareMedia. L'*AwarePhone* est une application qui fonctionne sur un Smartphone, qui permet d'afficher une liste de contacts pour un utilisateur et soutenir la messagerie des textes simples. Le contexte de travail pour chaque utilisateur est affiché sur la liste de contacts, permettant à l'utilisateur de maintenir la sensibilisation du contexte d'un collègue avant de lancer un appel. L'*AwarePhone* est intégré dans la fonctionnalité de l'appareil mobile (Chen and Kotz 2000).

1.6.3.2 Interface utilisateur adaptative et consciente au contexte

Les interfaces utilisateurs sensibles au contexte sont un cas particulier des fonctions sensibles au contexte. Un exemple très simple d'une interface utilisateur sensible au contexte est le rétro-éclairage d'un dispositif qui est mis en marche lorsque l'environnement est sombre. D'autres exemples sont des profils audio qui correspondent à une mise en page de la situation ou de l'écran particulier optimisés pour un contexte donné. Sur un appareil mobile, les modalités d'entrée peuvent dépendre du contexte (par exemple dans une voiture un appareil mobile peut utiliser un menu simple avec une grande police qui peut être activé par des commandes vocales simples (Chen and Kotz 2000).

1.6.3.3 Autres applications mobiles conscientes au contexte

D'autres applications sensibles aux contextes interrogés par Chen et Kotz (Chen and Kotz 2000) comprennent :

- Téléportation, un outil pour cartographier dynamiquement l'interface utilisateur sur les ressources des ordinateurs qui l'entourent;

- Assistant shopping, une application conçue pour guider les acheteurs dans un magasin, fournir des détails sur les articles, aider à localiser les articles, indiquer les articles en vente, et faire une analyse comparative des prix;
- Les systèmes mondiaux d'adaptation pour le téléphone mobile et les assistants numériques personnels, étaient profils du téléphone mobile est automatiquement sélectionné en fonction du contexte reconnu. Le téléphone choisit de sonner, vibrer, régler le volume de sonnerie ou de garder le silence, selon que le téléphone est dans la main, sur une table, dans une valise, ou à l'extérieur.
- Location-aware Informations Livraison, où chaque message de rappel est créé avec un emplacement; lorsque le destinataire prévu arrive à cet endroit, le message est envoyé par synthèse vocale, sans que l'utilisateur doive maintenir le dispositif et de le lire sur l'écran.

1.7 Discussion

D'après l'étude du domaine de Mobile Computing, nous avons remarqué que le développement des applications mobiles est fait de façon ad-hoc et aléatoire en se basant sur les critères posés par le langage d'implémentation utilisé, sans revenir à un modèle bien défini et standardisé. En d'autres termes, si nous positionnons les étapes de développement des applications mobiles dans le modèle de développement en cascade par exemple, nous remarquons que, la phase d'étude de faisabilité et la phase de conception (générale et détaillée) ne sont pas prises en considération.

Nous attestons que, dans le but de répondre aux besoins des utilisateurs toujours changeants, faire face à l'augmentation constante d'utilisation des applications mobiles et enfin, développer des applications mobiles plus performantes ; une discipline ou un modèle de développement est indispensable dans le processus de leurs développements.

1.8 Conclusion

Nous avons présenté dans ce chapitre, une partie du contexte global dans lequel s'intéresse cette thèse: le domaine des applications mobiles. Nous avons montré comment les appareils mobiles ont envahi notre vie quotidienne en présentant quelques statistiques récentes. Comme nous nous intéressons à la modélisation des applications mobiles à base de Cloud, nous arguons qu'il est nécessaire de passer par les applications mobiles traditionnelles. Pour cela nous avons présenté dans ce chapitre les notions liées au développement des applications mobile, commençant par leurs différents types et en terminant par les types de technologies utilisées pour leurs implémentations.

En passant par les applications mobiles traditionnelles, nous présentons dans le chapitre suivant le nouveau paradigme Mobile Cloud Computing et les notions concernant la nouvelle génération des applications mobiles (c.-à-d. Application mobile à Base de Cloud).

Chapitre 2

Introduction au paradigme Mobile Cloud Computing et les Applications Mobiles à Base de Cloud

2.1 Introduction

Être les moyens les plus efficaces, convenables, et les plus utilisés pour accéder à l'Internet, les appareils mobiles ont gagné une importance très considérable dans notre vie quotidienne. Les progrès rapides de Mobile Computing (MC) (Satyanarayanan 2010) deviennent une tendance puissante du développement des technologies de l'information ainsi que le commerce et l'industrie. Par contre, les périphériques mobiles sont confrontés à de nombreux défis dans leurs ressources (ex. la durée de vie de la batterie, capacité de stockage et bande passante) et communications (ex. mobilité et sécurité) (Satyanarayanan 1996). Les ressources limitées des appareils mobiles à entraver de manière significative à l'amélioration des qualités des services offerts par les applications mobiles.

Le Cloud Computing (CC) a été largement reconnu comme l'infrastructure informatique de la prochaine génération. Le CC offre certains avantages en permettant aux utilisateurs d'utiliser des services de type : Infrastructure (ex. serveurs, réseaux et base de données), plates-formes (ex. Des middlewares et des systèmes d'exploitation) et les logiciels (ex. applications). Les services sont fournis par des fournisseurs de Cloud (ex. Google, Amazon, et Salesforce) à faible coût. En outre, CC permet aux utilisateurs d'utiliser les ressources de façon élastique dans une manière sur demande. Par conséquent, les applications mobiles peuvent être provisionnées rapidement et publiées avec le minimum d'efforts pour leurs gestions et sans les interactions avec les fournisseurs de services.

Avec l'explosion des applications mobiles et le support de CC en termes de variété de services fournis pour les utilisateurs mobiles, le Mobile Cloud Computing (MCC) est présenté comme un entrelacement entre le Mobile Computing et le Cloud Computing. Le Mobile Cloud Computing apporte de nouveaux types de services et d'installations pour les utilisateurs mobiles afin d'exploiter pleinement les avantages du Cloud Computing.

Dans ce chapitre, nous présentons une étude exhaustive sur le paradigme Mobile Cloud Computing (MCC) y compris la définition, l'architecture et les avantages de MCC. Aussi nous allons donner un aperçu sur l'utilisation du MCC dans diverses applications et mettre l'accent sur les questions les plus posées dans ce domaine avec les approches qui essaient de les traiter.

2.2 Aperçu sur le Mobile Cloud Computing

Le terme « Mobile Cloud Computing » a été introduit pas longtemps après que le concept de « Cloud Computing » est apparu en 2007. Le MCC a attiré l'attention des entrepreneurs comme une option commerciale rentable qui réduit le développement et le coût d'exécution des applications mobiles, celle des utilisateurs comme une nouvelle technologie pour réaliser une expérience riche de toute une variété de services mobiles à faible coût, et enfin celle des chercheurs comme une solution idéale pour le Green IT (Ali 2009).

2.2.1 Définition

Le forum de Mobile Cloud Computing définit MCC comme suite (Forum 2016): Mobile Cloud Computing à sa plus simple expression, se réfère à une infrastructure où le stockage des données et le traitement de données surviennent en dehors de l'appareil mobile. Le MCC déplace la puissance de calcul et le stockage de données loin de téléphones mobiles et dans les Clouds.

Aepona (AEPONA November 2010.) décrit le MCC comme un nouveau paradigme où le traitement et stockage des données sont déplacé de l'appareil mobile à une plateforme puissante, centralisée et située dans le Cloud. Ces applications centralisées sont ensuite accessibles via la connexion sans fil basée sur un mince client natif ou un navigateur web sur les appareils mobiles.

Brièvement, le MCC fournit aux utilisateurs mobiles des services de traitement et de stockage de données dans les Clouds. Les appareils mobiles n'ont pas besoin d'une configuration puissante (ex. la vitesse de CPU et une grande capacité de stockage) puisque tous les modules complexes en termes de gestion peuvent être traités dans le Cloud.

2.2.2 Architecture du mobile Cloud Computing

Comme définis précédemment, le MCC combine le Cloud et le mobile, par conséquent l'architecture proposée pour ce paradigme comporte les concepts de base du Cloud et ceux du mobile. L'ensemble de ces concepts ainsi que le schéma général de leurs interactions sont présentés dans la figure 2.1. Les appareils mobiles sont connectés à un réseau via des stations de base *Base Transciever Station (BTS), satellite et Access Point* qui établissent et contrôlent les connexions et les interfaces fonctionnelles entre le mobile et les réseaux. Les demandes et les informations des utilisateurs (ex. Leurs localisations et leurs ID) sont transmises au Central processor qui est connecté au serveur fournisseur de service mobile. En se basant sur *Home Agent* (HA) et *Data base*, le mobile *Network Operators* fournit aux utilisateurs des services mobiles comme l'autorisation et l'authentification, ensuite les demandes sont envoyées au Cloud via l'Internet. Au niveau du Cloud le contrôleur *Cloud Controller* fournit à l'appareil mobile les services Cloud demandés. Ces services sont conçus avec le concept d'informatique utilitaire (c.-à-d. Utility Computing), Virtualisations et architecture orientée service (ex. Application web, serveur et base de donnée).

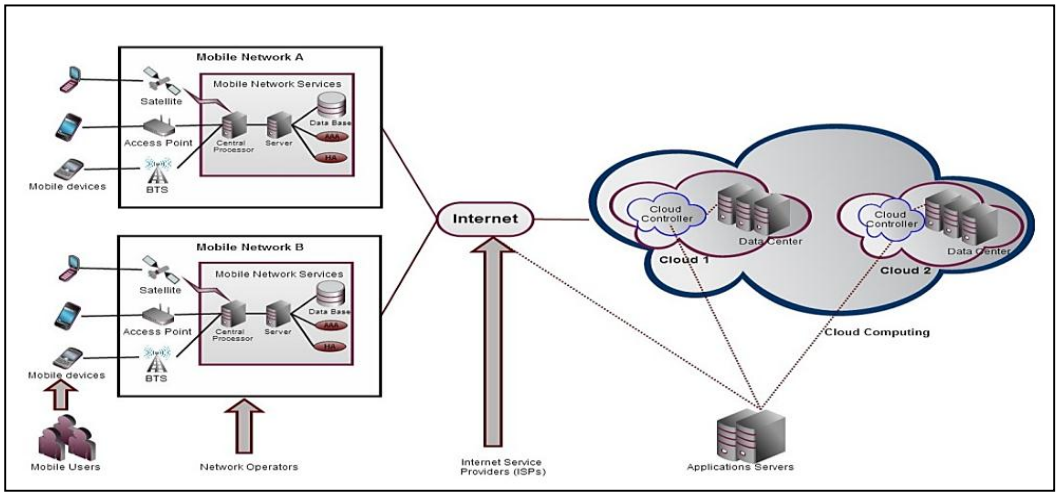


Figure 2. 1 L'architecture Mobile Cloud Computing (Dinh *et al.* 2011).

Le MCC est le résultat de la synergie entre le CC et le MC. Alors, il est fondamental de passer par la définition de quelques concepts très importants dans le domaine du CC avant d'aborder ceux du MCC.

2.3 Aperçu sur le Cloud Computing

Les détails de l'architecture Cloud pourraient être différents dans des contextes différents. Par exemple, une architecture à quartes couches est expliquée dans (Liu *et al.* 2010) pour comparer le Cloud Computing avec le Grid Computing. Alternativement, une architecture orientée service, appelé Aneka est introduite pour permettre aux développeurs de créer des applications .NET avec la prise en charge des interfaces de programmation des applications (API) et les modèles de programmation multiples (Vecchiola *et al.* 2009). R.Buyya dans (Buyya *et al.* 2009) présente une architecture pour la création de Cloud orientée vers le marché, et Y.Huang dans (Huang 2011), propose une architecture des services d'affaires web livrés. Dans ce chapitre, nous nous concentrons sur une architecture multicouche du Cloud Computing (Figure2.2). Cette architecture est communément utilisée pour démontrer l'efficacité du modèle de « Cloud Computing » en termes de satisfaction de besoins de l'utilisateur (Tsai *et al.* 2010).

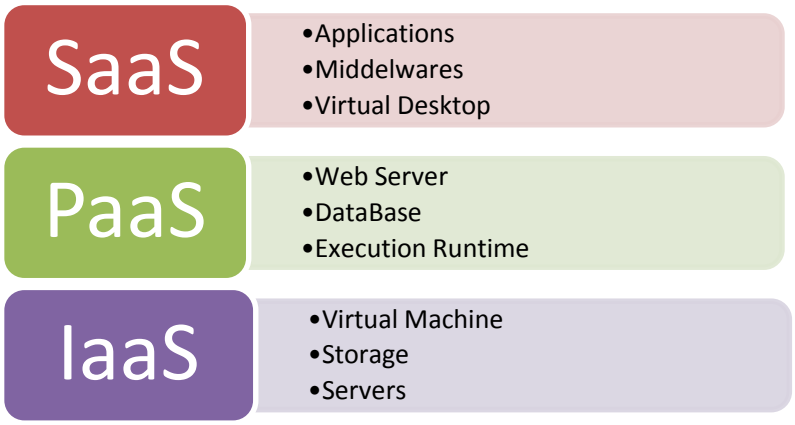


Figure 2. 2 Architecture Orienté Service du Cloud Computing

Généralement, le Cloud Computing est un système de réseau distribué à grande échelle. Ce système est basé sur un certain nombre de serveurs situés dans des centres de données. Les services Cloud sont généralement classés en se basant sur un concept de couche (Figure2.2). Dans les couches inférieures de ce paradigme, on trouve que l'infrastructure en tant que service (IaaS), plateforme en tant que service (PaaS), et le logiciel en tant que service (SaaS) sont empilées.

2.3.1 Couche Centre de Données

Cette couche implicite fournit le matériel d'installation et l'infrastructure pour les Clouds. Dans la couche du centre de données, un certain nombre de serveurs sont reliés avec des réseaux à haute vitesse afin de fournir des services pour les clients. Typiquement, les centres de données sont construits dans les régions les moins habitées, avec une haute stabilité d'alimentation et un faible risque de catastrophe.

2.3.2 Infrastructure en tant que service (IaaS)

L'IaaS, permet l'approvisionnement d'espace de stockage, le matériel, les serveurs et les composants réseau. Le client paie généralement sur l'utilisation. Ainsi, les clients peuvent économiser le coût comme le paiement est uniquement basé sur la quantité de ressources qu'ils utilisent vraiment. L'infrastructure peut être élargie ou rétrécie de manière dynamique selon les besoins. Quelques exemples d'IaaS sont : Amazon EC2 (Elastic Cloud Computing) et S3 (Service de stockage simple).

2.3.3 Plateforme en tant que service (PaaS)

La PaaS, offre un environnement intégré avancé pour la création, le test et le déploiement des applications personnalisées. Les exemples de PaaS sont Google App Engine, Microsoft Azure et Amazon Map Reduce/Service de stockage simple.

2.3.4 Logiciels en tant que service (SaaS)

SaaS prend en charge la distribution des logiciels selon des exigences spécifiques. Dans cette couche, les utilisateurs peuvent accéder à des applications et des informations à distance via l'Internet et ne payer que pour ce qu'ils utilisent. Salesforce est l'un des pionniers en fournissant ce modèle de service. Microsoft Live Mesh permet également le partage de fichiers et de dossiers entre plusieurs périphériques simultanément.

Bien que l'architecture de Cloud Computing puisse être divisée en quatre/trois couches comme montrées dans la figure2.2, cela ne veut pas dire que la couche supérieure doit être construite sur la couche directement inférieure. Par exemple, l'application SaaS peut être déployée directement sur l'IaaS, au lieu du PaaS. En outre, certains services peuvent être considérés comme une partie de plus d'une couche. Par exemple, le service de stockage de données peut être considéré comme l'un ou l'autre de l'IaaS ou PaaS. Selon ce modèle architectural, les utilisateurs peuvent utiliser les services de manière souple et efficace.

2.3.5 Modèles de Cloud

Il est à noter que, selon la politique d'approvisionnement des services définis par les fournisseurs, il existe quatre modèles de déploiement de Cloud comme montré dans la figure 2.3.

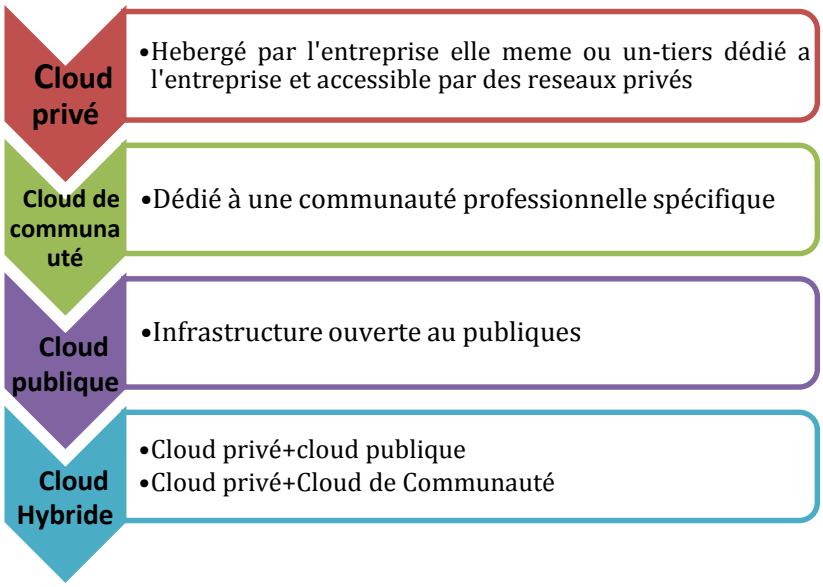


Figure 2. 3 Les modelés de déploiement du Cloud.

2.3.5.1 Cloud privé

Le Cloud privé est la typologie du Cloud la plus répandue; 73% des Clouds déployés dans les entreprises sont des Clouds privés selon Gartner.

Cette infrastructure Cloud est gérée par l'organisation ou à un tiers et exploité uniquement pour les besoins de l'organisation. Cela peut exister sous ou hors tension prémisse. Un exemple de ceci est *Redmine*, qui utilise sa propre installation VMware VCloud pour déployer son système. Ce modèle de Cloud peut se déployer sous deux formes distinctes :

– *Cloud privé interne* : les données sont hébergées par l'entreprise elle-même, parfois partagées ou mutualisées en mode privatif avec les filiales.

– *Cloud privé externe* : les données sont hébergées chez un tiers, il est entièrement dédié à l'entreprise et accessible via des réseaux sécurisés du type VPN.

2.3.5.2 Cloud communautaire

Le Cloud communautaire, comme son l'indique est utilisé par plusieurs organisations qui ont des besoins communs. Le Cloud communautaire, peut héberger une application métier très spécialisé, mais commun à plusieurs entités, qui décident de fédérer leurs efforts en construisant un Cloud pour l'héberger et la gérer.

2.3.5.3 Cloud public

Le Cloud public est accessible depuis Internet et géré par un prestataire externe ; l'avantage de ce Cloud est que les ressources peuvent être partagées entre plusieurs entreprises, parfois même concurrentes.

2.3.5.4 Cloud hybride

Cette infrastructure Cloud est composée de deux ou plusieurs types de Cloud énumérés ci-dessus qui restent des entités uniques, mais sont connectés via la technologie standardisée qui permet la portabilité des données et des applications.

2.4 Avantages du Mobile Cloud Computing

Le MCC est connue comme une solution prometteuse pour le Mobile Computing pour plusieurs raisons (ex. la mobilité, la communication et la portabilité (Forman and Zahorjan 1994). Dans ce qui suit, nous décrivons comment le Cloud Computing peut être utilisé pour surmonter les obstacles dans le domaine de l'informatique mobile, ce qui en soulignant les avantages du MCC.

2.4.1 Prolonger la durée de vie de la batterie

La batterie est l'une des principales préoccupations pour les périphériques mobiles. Plusieurs solutions ont été proposées pour améliorer les performances du processeur (Kakerow 2002, Paulson 2003) et gérer le disque et l'écran de façon intelligente (Davis 1993), (Mayo and Ranganathan 2003), afin de réduire la consommation d'énergie. Toutefois, ces solutions nécessitent des changements dans la structure des appareils mobiles, ou ils ont besoin d'un nouveau matériel qui entraîne une augmentation des coûts et peut ne pas être possible pour tous les périphériques mobiles.

La technique de déchargement (c.-à-d. Offloading) est proposé avec l'objectif de migrer les grands calculs et les traitements complexes depuis le périphérique limité en ressources (c'est-à-dire les périphériques mobiles) pour une machine ingénieuse (c'est-à-dire, les serveurs dans les Clouds). Cela n'évitera que les applications mobiles de prennent un long temps d'exécution sur les appareils mobiles, ce qui entraîne de grandes quantités de consommation d'énergie.

Rudenko et *al.* dans (Rudenko *et al.* 1998) et A Smailagic et *al.* dans (Smailagic and Ettus 2002) évaluent l'efficacité des techniques de déchargement par le biais de plusieurs expériences. Les résultats démontrent que l'exécution des applications à distance peut économiser l'énergie de façon significative.

En outre, nombreuses applications mobiles profitent de la tâche de migration et de traitement à distance. Par exemple, en déchargeant une optimisation du compilateur pour le traitement d'images (Kremer *et al.* 2001) peut réduire de 41 % de consommation d'énergie d'un appareil mobile. Aussi, l'utilisation Memory Unit Arithmétique Interface (MAUI) pour déplacer les composants de jeu mobile (Cuervo *et al.* 2010) aux serveurs

dans le Cloud peut enregistrer 27 % de la consommation d'énergie pour les jeux d'ordinateurs et 45 % pour le jeu d'échecs.

2.4.2 Améliorer la capacité de stockage et la puissance de traitement

Capacité de stockage est aussi une contrainte pour les périphériques mobiles. Le MCC est mis au œuvre pour permettre aux utilisateurs mobiles de sauvegarder/accéder aux données sur le Cloud via les réseaux sans fil.

Le premier exemple est le service de stockage simple d'Amazon (Amazon S3) (AwS3 2016), qui prend en charge File Storage Service. Un autre exemple est Image Exchange qui utilise le grand espace de stockage dans les Clouds pour les utilisateurs mobiles (Vartiainen and Väänänen-Vainio-Mattila 2010). Ce service de partage de photo permet aux utilisateurs mobiles de sauvegarder des images sur les Clouds immédiatement après leurs captures. Les utilisateurs peuvent accéder à toutes les images à partir de tous les périphériques. Avec le Cloud, les utilisateurs peuvent économiser beaucoup d'énergie et d'espace de stockage sur leurs appareils mobiles depuis toutes les images sont envoyées et traitées sur les Clouds. Flickr (Flickr 2016) et ShoZu (Shozu 2016) sont également le succès des applications de partage de photos mobile basé sur MCC.

Le MCC aide à réduire le coût d'exécution des applications intensives qui consomment un temps et énergie considérable en s'exécutant sur des appareils mobiles. Le Cloud Computing peut soutenir efficacement diverses tâches pour l'entreposage de données (c.-à-d. Data Warehousing), la gestion et la synchronisation de plusieurs documents en ligne. Par exemple, les Clouds peuvent être utilisés pour le transcodage (Garcia and Kalva 2011), jeux d'échecs (Cuervo *et al.* 2010), ou la radiodiffusion de services multimédia (Li *et al.* 2001) pour les périphériques mobiles.

2.4.3 Améliorer la fiabilité

Le stockage de données ou l'exécution des applications sur les Clouds est un moyen efficace d'améliorer la fiabilité. Ceci réduit le risque de perte de données et d'application sur les appareils mobiles. En outre, le MCC peut être conçu comme un modèle de sécurité pour les fournisseurs de services et les utilisateurs. Par exemple, le Cloud Computing peut être utilisé pour protéger le contenu numérique protégé par le droit d'auteur (ex. vidéo, le clip et la musique) contre la maltraitance et la distribution non autorisée (Zou *et al.* 2010). Aussi, le Cloud peut fournir aux utilisateurs mobiles des services de sécurité tels que l'analyse antivirus, la détection de code malveillant, et Authentification(Oberheide *et al.* 2008). Les services de sécurité basés sur le Cloud peuvent faire un usage efficace de la collection d'enregistrement de différents utilisateurs pour améliorer l'efficacité des services.

En outre, le MCC hérite également certains avantages des Clouds pour les services mobiles comme suit :

2.4.3.1 Provisionnement dynamique

Le provisionnement dynamique, à la demande des ressources sur une fine granularité, à base de self-service est un moyen flexible pour les fournisseurs de services et les utilisateurs mobiles pour exécuter leurs applications sans réservation avancée des ressources.

2.4.3.2 Évolutivité

Le déploiement des applications mobiles peut être effectué et adapté pour répondre aux exigences imprévisibles de l'utilisateur. Les fournisseurs de services peuvent facilement ajouter et développer une application et un service sans ou avec peu de contraintes sur l'utilisation des ressources.

2.4.3.3 Localisations multiples

Les fournisseurs de services (ex. opérateur réseaux et centre de données) peuvent partager les ressources et les coûts nécessaires pour financer une variété d'application et un grand nombre d'utilisateurs.

2.4.3.4 Facilité d'intégration

Plusieurs services de différents fournisseurs peuvent être intégrés facilement à travers les Clouds et l'Internet pour répondre aux demandes des utilisateurs.

2.5 Applications Mobiles à Base de Cloud

Les applications mobiles gagnent une partie croissante dans un marché mobile mondial. Diverses applications mobiles ont pris les avantages du MCC. Dans cette section, certaines applications typiques du MCC sont introduites (Dinh *et al.* 2011).

2.5.1 Mobile Commerce

Le commerce mobile (m-commerce) est un modèle d'affaires pour le commerce à l'aide de périphériques mobiles. Les applications de m-commerce généralement réalisent certaines tâches qui nécessitent une mobilité (ex. Transaction et payement mobile, la messagerie mobile et billetterie mobile).

Les applications de m-commerce peuvent être classées dans un petit nombre de catégories, notamment les finances, la publicité et le magasinage (Tableau 2.1). Les applications m-commerce ont à faire face à divers défis (ex. faible bande passante réseau, haute complexité de configurations de périphérique mobile, et la sécurité). Par conséquent, les applications de m-commerce applications sont intégrées à l'environnement de Cloud Computing pour aborder ces questions.

Classe d'applications	Type	Exemple
Application de m-Financial	B2C, B2B	Banques, sociétés de courtage, frais d'utilisateur mobile
Application de m-Advertising	B2C	Envoi personnalisé de la publicité selon l'emplacement physique de l'utilisateur
Application de m-Shopping	B2C, B2B	Repérez / commander certains produits à partir d'un terminal mobile

Table 2. 1 Les Classes des Application des M-Commerce (Dinh *et al.* 2011).

X.Yang et *al*. proposent dans (Yang *et al.* 2010) une plateforme 3G de commerce électronique basée sur le Cloud Computing. Ce paradigme allie les avantages de réseau 3G et le Cloud Computing pour augmenter la vitesse de traitement des données et le niveau de sécurité (Dai *et al.* 2007) en se fondant sur l'infrastructure de clés publiques (*Public Key Infrastructure*). Le mécanisme de l'PKI utilise une clé de chiffrement et de contrôle d'accès basé sur un cryptage pour garantir la confidentialité de l'utilisateur en accédant à l'impartition des données. Dans (Leina *et al.* 2010), Z. Leina et *al*. proposent un *4PL-AVE Trading Platform* qui utilise le Cloud Computing afin d'accroître la sécurité pour les utilisateurs et améliorer leurs satisfactions, l'intimité client, et la compétitivité des coûts.

2.5.2 Mobile Learning

L'apprentissage mobile (m-learning) est conçu en se basant sur l'apprentissage électronique (e-learning) et la mobilité. Toutefois, les applications m-learning traditionnelles ont des limitations en termes de coût élevé des périphériques et du réseau, faible taux de transmission réseau, et ressources pédagogiques limitées (Chen *et al.* 2010), (Gao and Zhai 2010), (Li 2010). Reposant sur le Cloud Computing, les applications de m-learning sont proposées pour résoudre ces limitations. Par exemple, en utilisant un Cloud avec une grande capacité de stockage et puissance de traitement, les applications fournissent aux apprenants des services beaucoup plus riches en termes de taille de données (information), vitesse de traitement plus rapide et prolonger la durée de vie de la batterie.

W. Zhao et *al*. présentent dans (Zhao *et al.* 2010) les avantages de l'association du m-learning avec le Cloud Computing pour améliorer la qualité de la communication entre les étudiants et les enseignants. Par l'intermédiaire d'un site web construit sur Google Apps Engine, les étudiants communiquent avec leurs enseignants à tout moment. En outre, les enseignants peuvent obtenir de l'information sur le niveau de connaissances des étudiants concernant les cours et peuvent répondre aux questions des élèves en temps opportun. En plus, un système de m-learning contextuelle basée sur plateforme IMERA (Yin *et al.* 2009) indique qu'un système de m-learning basé Cloud aide les apprenants à avoir accès à des ressources d'apprentissage à distance.

Un autre exemple des applications mobiles basées Cloud dans l'apprentissage est « *Cornucopia* ». Cette application est mise en œuvre pour les recherches d'étudiants

premiers cycle de génétique et « *plantations Pathfinder* » conçu pour fournir de l'information et fournissent un espace de collaboration pour les visiteurs lorsqu'ils visitent les jardins (Rieger and Gay 1997).

L'objectif du déploiement de ces applications est d'aider les étudiants à améliorer leurs compréhensions de la conception appropriée de MCC en soutenant des expériences vécues sur le terrain.

Dans (Ferzli and Khalife 2011), R.Frezli et *al*. Proposent un outil d'éducation implémenté en se basant sur le Cloud Computing pour créer un cours sur le traitement image/vidéo. Par le biais d'appareils mobiles, les apprenants peuvent comprendre et comparer différents algorithmes utilisés dans les applications mobiles (ex. détection de visage, et l'amélioration de l'image).

2.5.3 Mobile HealthCare

Le but des applications mobiles à base de Cloud dans le domaine médical est de minimiser les limites de traitement médical traditionnel (ex. capacité de stockage insuffisante, la sécurité et la protection de la vie privée, et les erreurs médicales (Kohn *et al.* 2000), (Kopec *et al.* 2003)). Le m-healthcare offre aux utilisateurs mobiles un accès facile et rapide aux ressources (ex. dossiers médicaux). En outre, m-healthcare propose aux hôpitaux et les organismes de santé une variété de services à la demande sur les Clouds plutôt que l'acquisition des applications autonomes sur des serveurs locaux.

Il existe quelques régimes d'application mobile à base de Cloud dans le secteur de la santé, par exemple, U.Varshney et *al.* présentent cinq applications principales de m-healthcare dans l'environnement omniprésent (Varshney 2007).

- *Services de surveillance de santé globale* : Ces Services permettent aux patients d'être surveillés à tout moment et en tous lieux par l'intermédiaire de communications sans fil.
- *Système de gestion d'urgence intelligent* : Le système proposé peut gérer et coordonner efficacement et rapidement l'ensemble de la flotte de véhicules d'urgence lors de reçoit des appels d'accidents ou d'incidents.
- *Appareils mobiles conscients à la santé :* Ce genre d'appareil détecte les taux d'impulsion, la pression artérielle, et le niveau de l'alcool pour avertir le système d'urgence de santé.
- *L'accès ubiquitaire a l'information d'healthcare :* ce système permet aux patients ou aux fournisseurs de soins de santé d'accéder aux informations médicales actuelles et passées.
- *La gestion de l'incitation Life style omniprésente :* Ce genre d'application peut être utilisé pour payer les dépenses de soins de santé et de gérer d'autres frais connexes automatiquement.

De même, C. Douka et *al.* proposent @HealthCloud (Doukas *et al.* 2010), un prototype de m-healthcare information management système basé sur le Cloud et un client mobile

fonctionnant sous Android. Ce prototype présente trois services utilisant le service de stockage S3 pour gérer les dossiers de santé du patient et les images médicales.

Pour les systèmes pratiques, un système de gestion des soins à domicile de télémédecine est mis en œuvre à Taïwan pour surveiller les participants, en particulier pour les patients souffrant d'hypertension et de diabète. Le système surveille 300 participants et stocke plus de 4736 enregistrements de la pression artérielle et les données de mesure de sucre sur le Cloud. Lorsqu'un participant effectue le glucose sanguin/mesure de pression via l'équipement spécialisé, l'équipement peut envoyer les paramètres mesurés automatiquement dans le système, ou le participant peut envoyer des paramètres par SMS via leurs périphériques mobiles. Après cela, le Cloud va recueillir et analyser l'information sur le participant et renvoyée des résultats (Tang *et al.* 2010).

Le développement du mobile healthcare fournit d'énormes aides aux participants. Toutefois, les informations qui doivent être recueillies et gérées liées aux renseignements personnels sur la santé sont sensibles. Par conséquent, dans (Hoang and Chen 2010), (Nkosi and Mekuria 2010) proposent des solutions afin de protéger l'information sur la santé du participant, augmentant ainsi la confidentialité des services. Tandis que Hoang et Chen utilisent P2P pour fédérer les Clouds afin de répondre aux questions de sécurité, protection des données. Le modèle proposé dans (Nkosi and Mekuria 2010) fournit la sécurité comme un service sur le Cloud pour protéger des applications mobiles. Par conséquent, les fournisseurs des applications mobiles de soins de santé et les utilisateurs n'auront pas à se soucier de la sécurité puisqu'elle est assurée par du fournisseur de sécurité.

2.5.4 Mobile Gaming

Le m-jeux est un marché potentiel qui génère un revenu pour les fournisseurs de services. M-Jeu peut complètement décharger le moteur de jeu exigeant une grande ressource de calcul (ex. le rendu graphique) au serveur dans le Cloud où les joueurs uniquement interagissent avec l'interface sur l'écran de leurs appareils.

Z. Li et *al.* en (Li and Hua 2010) démontrent que le déchargement (code multimédia) peut économiser de l'énergie pour les appareils mobiles, ce qui augmente le temps de.

S. Wang et *al.* (Li *et al.* 2001) présentent un nouveau m-jeu reposant sur le Cloud, en utilisant technique d'adaptation de déchirement pour ajuster dynamiquement les paramètres de rendue du jeu selon les contraintes de communication et les exigences des joueurs. La technique d'adaptation de déchirement principalement repose sur l'idée de réduire le nombre d'objets dans la liste d'affichage puisque tous les objets dans la liste d'affichage créé par moteur de jeu sont nécessaires pour jouer le jeu et intensifier la complexité des opérations de déchirement. L'objectif est d'optimiser l'expérience de l'utilisateur étant donné les coûts de traitement et des communications.

2.5.5 Autres applications mobiles Cloud pratiques

Un Cloud devient un outil utile pour aider les utilisateurs mobiles de partager des photos et des clips vidéo efficacement et marquer leurs amis dans les réseaux sociaux les plus populaires, comme Twitter et Facebook.

MeLog (Li and Hua 2010) est une application mobile à base de Cloud qui permet aux utilisateurs mobiles de partager l'expérience en temps réel (ex. voyage, shopping, et événement) au-dessus des Clouds grâce à un système automatique de blogs. Les utilisateurs mobiles (ex. les voyageurs) sont soutenus par plusieurs services de Cloud tels que guider leur voyage, fournir des cartes géographiques, et stockage d'images et de vidéo.

Z. Ye introduit un service de localisation mobile permettant aux utilisateurs de capturer une courte vidéo sur les bâtiments dans leurs entourages. L'algorithme de comparaison s'exécute au Cloud pour utiliser une grande quantité de données afin de chercher l'emplacement de ces bâtiments (Ye *et al.* 2010).

En outre, une *Hour-Traduction* fournit un service de traduction en ligne en cours d'exécution sur le Cloud d'Amazon Web Services. Cette application permet aux utilisateurs mobiles, en particulier les touristes, de recevoir l'information traduite dans leurs langues, via leurs appareils mobiles(onehourtranslation 2016) .

En plus, il y a une application mobile collaborative basée Cloud (Angin *et al.* 2010) qui détecte les feux de circulation pour les aveugles. Aussi, un Framework basé Cloud (Li *et al.* 2010) pour surveiller différents coins dans une maison au moyen d'un appareil mobile, et certains efforts qui intègrent les services actuels (ex. BitTorrent et le réseau social mobile) dans les Clouds comme dans (Zhenyu *et al.* 2010), (Kelényi and Nurminen 2010).

2.6 Approches de Mobile Cloud Computing

Tel que discuté dans la section (2.5), le MCC a de nombreux avantages pour les utilisateurs mobiles et les fournisseurs de services. Toutefois, en raison de l'intégration de deux domaines différents, c'est-à-dire, le Cloud Computing et Mobile Computing, le MCC doit affronter de nombreux défis de nature technique. Cette section répertorie plusieurs questions de recherche dans les MCC, qui sont liés à la communication mobile et le Cloud Computing. Puis, les solutions disponibles pour traiter de ces questions sont examinées.

2.6.1 Questions de côté communication

2.6.1.1 Une faible bande passante

La bande passante est l'un des gros problèmes dans les MCC depuis la ressource radio pour réseaux sans fil est bien maigres en comparaison avec les réseaux filaires traditionnels.

X. Jin et *al*. (Jin and Kwok 2010) proposent une solution pour partager la bande passante limitée entre les utilisateurs mobiles qui sont situés dans la même région (ex. un

milieu de travail, une station ou un stade) et impliqués dans le même contenu (ex. un fichier vidéo). Les auteurs modélisent l'interaction entre les utilisateurs comme un jeu de coalition. Par exemple, les utilisateurs forment une coalition où chaque membre est responsable d'une partie des fichiers vidéo (ex. de sons, d'images ou légendes) et transmet/échanges à d'autres membres de la coalition. Cela résulte une amélioration de la qualité de vidéo. Toutefois, la solution proposée est appliquée seulement dans le cas lorsque les utilisateurs dans une certaine zone sont intéressés par le même contenu. Aussi, il ne considère pas une politique de distribution (ex. qui reçoit combien et quelle partie des matières) qui conduit à un manque d'équité de la contribution de chaque utilisateur à une coalition.

E. Jung et *al.* considèrent la politique de distribution des données qui détermine quand et combien de portions de la bande passante disponible sont partagées entre les utilisateurs à partir de quels réseaux (ex. Wifi et WiMAX). Ils recueillent des profils d'utilisateur (par exemple, profile d'appels, profile d'intensité du signal, et profile d'alimentation) périodiquement et créent les tables de décision à l'aide de l'algorithme de processus de décision markoviens (MDP). Basé sur les tables, les utilisateurs ont le choix de décider ou non , aider les autres utilisateurs à télécharger des contenus qu'ils ne peuvent pas recevoir par eux-mêmes en raison de la limitation de bande passante, et dans quelle mesure il devrait aider (ex. 10 % de contenu). Les auteurs ont construit un Framework, appelé RACE (*Resource-Aware Collaborative Execution*), sur le Cloud pour bénéficier des avantages de ressources informatiques pour le maintien des profils utilisateur. Cette approche est adaptée aux utilisateurs qui partagent une bande passante limitée, pour équilibrer le rapport entre les avantages de l'assistance et les coûts d'énergie (Jung *et al.* 2010).

2.6.1.2 La Disponibilité

La disponibilité du service devient de plus en plus un problème important dans le MCC que dans le Cloud Computing avec les réseaux câblés. Les utilisateurs mobiles peuvent ne pas être en mesure de se connecter à Internet pour obtenir le service en raison de la congestion de la circulation, les pannes de réseau, et l'out-of-signal.

Dans (Huerta-Canepa and Lee 2010), les auteurs décrivent un mécanisme de découvert pour rechercher les nœuds dans le voisinage d'un utilisateur dont le lien vers le Cloud est indisponible. Après la détection de nœuds à proximité qui sont en mode stable, le fournisseur de cible pour l'application est modifié. De cette manière, au lieu d'avoir un lien direct vers le Cloud, utilisateur mobile peut se connecter à Internet grâce à des nœuds voisins de façon ad hoc. Toutefois, elle ne considère pas la mobilité, la capacité de périphériques, et la protection de la vie privée des nœuds voisins.

Par contre les auteurs dans (Zhang *et al.* 2010) tentent de surmonter les inconvénients de (Huerta-Canepa and Lee 2010). En particulier, ils proposent un WiFi basé sur les systèmes multi-sauts de réseau appelé MoNet et un protocole de partage de contenus distribués pour la situation sans aucune infrastructure. Contrairement à (Zhang *et al.* 2010), cette solution estime de déplacer des nœuds dans la proximité de l'utilisateur. Chaque nœud diffuse périodiquement des messages de contrôle pour informer d'autres

nœuds de son état (ex. connectivité et paramétrage) et les mises à jour de contenu local. Selon les messages, chaque nœud maintient une liste de nœuds voisins et une liste de contenu et les estimations des niveaux d'autres stations de rôle basé sur l'espace disque, de bande passante et de l'alimentation. Puis, les nœuds avec le plus court saut en termes de longueur de chemin et ayant le rôle de plus hauts niveaux sont sélectionnés comme les nœuds intermédiaires pour recevoir le contenu. En outre, les auteurs examinent également les questions de sécurité pour les clients mobiles lorsqu'ils partagent des informations à l'aide de clé de compte (pour authentifier et crypter le contenu privé), clé ami (pour un canal sécurisé entre deux amis), et clé de contenu (pour protéger un contrôle d'accès).

Cette approche est beaucoup plus efficace que les systèmes de réseau social actuel, en particulier dans le cas d'un débranchement.

2.6.1.3 Hétérogénéité

MCC sera utilisé dans des réseaux très hétérogènes en termes d'interfaces de réseau sans fil. Différents nœuds mobiles accèdent à Internet grâce à différentes technologies d'accès radio comme WCDMA, GPRS, WiMAX, CDMA2000 et WLAN. Par conséquent, une question de comment gérer la connectivité sans fil tout en satisfaisant les exigences du MCC se pose (par exemple, une connectivité toujours disponible, une évolutivité à la demande de la connectivité sans fil, et l'efficacité énergétique des appareils mobiles).

A.Klein et *al.* (Klein *et al.* 2010) proposent une architecture pour fournir aux utilisateurs mobiles une stratégie intelligente d'accès au réseau, afin de répondre aux besoins de l'application. Cette architecture est fondée sur un concept d'*Intelligent Network Radio Acess* (IRNA) (Mannweiler *et al.* 2009) . IRNA est un modèle efficace pour faire face à la dynamique et l'hétérogénéité des réseaux d'accès disponibles. Pour appliquer IRNA dans le domaine du MCC, les auteurs proposent une architecture de gestion de contexte (AMC) dans le but d'acquérir, de gérer et de distribuer une information de contexte. Comme illustré dans la Figure 2.4, cette architecture est constituée de trois composants principaux : fournisseur de contexte (*Context Provider),* consommateur de contexte (*Context Consumer*), et le *Context Broker.* Cependant, le *Context Quality Enabler* est également nécessaire pour faciliter les opérations des autres composants. Dans cette architecture, lorsqu'un consommateur de contexte veut communiquer avec un fournisseur de contexte, le consommateur devra demander l'URI (Uniform Resource Identifier) de fournisseurs de contexte à l'égard du *Context Broker*. En utilisant cet URI, le consommateur peut communiquer directement avec le fournisseur et demande les données contextuelles. Par conséquent, ce processus augmente la vitesse de livraison de données contextuelles. En plus, lorsque le *Contexte Quality Enabler* reçoit les exigences concernant la qualité du contexte depuis le consommateur, il filtra l'URI de fournisseurs de contexte qui ne sont pas compatibles avec le niveau de qualité requis. Par conséquent,

cette architecture permet de contrôler la qualité de contexte selon les exigences des consommateurs.

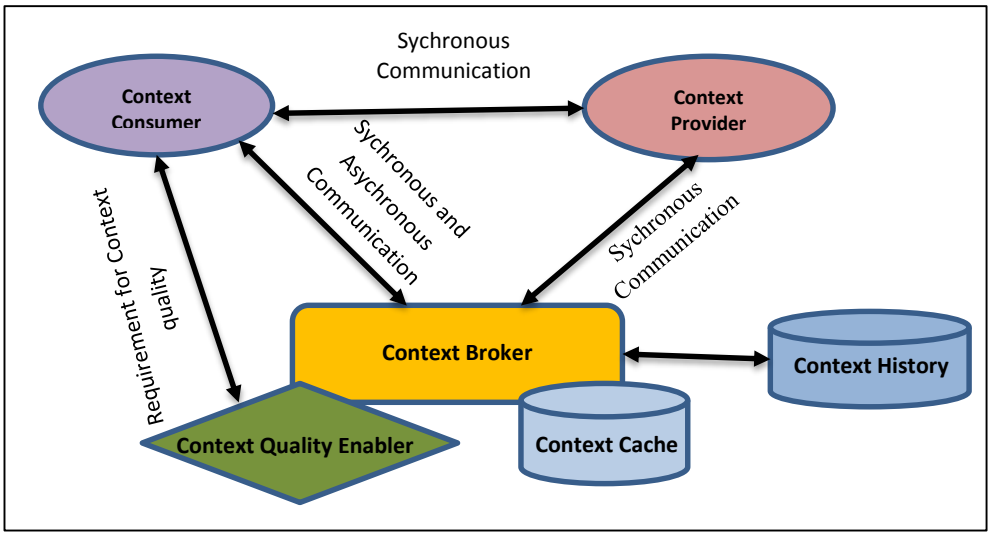


Figure 2. 4 Architecture de Gestion de contexte (Klein *et al.* 2010) .

2.6.2 Questions de côté Informatique

2.6.2.1 Le déchargement (*Offloading*)

Le déchargement est l'une des principales caractéristiques du MCC afin d'améliorer la durée de vie de la batterie pour les appareils mobiles et augmenter les performances des applications. Toutefois, il existe de nombreuses questions connexes, concernant l'efficacité et la dynamique du déchargement en fonction de changement d'environnement.

a) Le déchargement dans l'environnement statique

Les expériences dans (Rudenko *et al.* 1998) montrent que le déchargement n'est pas toujours la bonne façon d'économiser de l'énergie. Pour une compilation de code, le déchargement peut consommer plus d'énergie que celle de traitement local lorsque la taille des codes n'est pas petite. Par exemple, lorsque la taille de l'altération de codes après la compilation est 500KB, le déchargement consomme environ 5 % de la batterie d'un appareil, tandis que le traitement local consomme environ 10 % de la batterie pour son calcul. Dans ce cas, le déchargement peut gagner de l'énergie de la batterie jusqu'à 50 %. Toutefois, lorsque la taille de l'altération de codes est 250KB, l'efficacité réduit à 30 %. Lorsque la taille de l'altération de codes est petite, le déchargement consomme plus que celui de traitement local. Comme autre exemple (Rudenko *et al.* 1998) montre l'application de gaussienne (pour résoudre un système d'équations algébriques linéaires) qui décharge l'ensemble de la matrice dans le serveur distant. En termes de l'efficacité énergétique, le coût du déchargement est plus élevé pour les petites matrices (ex., inférieur à 500x500 en format) tandis que l'économie de coût peut être jusqu'à 45 % pour grandes matrices. Par conséquent, c'est un problème critique pour les appareils mobiles de déterminer s'il faut décharger et quelles parties de l'application de codes doivent être déchargées pour améliorer l'efficacité énergétique. En outre, différentes technologies

d'accès sans fil utilisent différentes quantités d'énergie et prennent en charge différents taux de transfert des données. Ces facteurs doivent être pris en compte.

K. Kumar et *al.* (Kumar and Lu 2010) proposent un programme de partitionnement basé sur l'estimation de la consommation d'énergie avant l'exécution du programme. Le programme optimal pour le partitionnement est calculé sur la base du compromis entre la communication et le calcul des coûts. Le coût de communication dépend de la taille des données transmises et de la bande passante réseau, tandis que le coût de calcul est touché par le temps de calcul. Toutefois, des informations telles que les exigences de communication ou le calcul de la charge de travail peut changer dans différentes instances d'exécution. Ainsi, des décisions optimales d'un programme le partitionnement doit être fait au moment de l'exécution de manière dynamique.

Plusieurs solutions sont proposées pour trouver la décision optimale pour les applications de partitionnement avant le déchargement.

Dans (Wang and Li 2004), les auteurs présentent une approche pour décider quels éléments des programmes Java doivent être déchargés. Cette approche d'abord divise un programme Java en méthodes et utilise des paramètres d'entrée (ex. taille des méthodes) pour calculer les coûts d'exécution pour ces méthodes. Ensuite, cette méthode compare les coûts d'exécution locale de chaque méthode avec les coûts d'exécution à distance qui sont estimés en fonction de l'état de l'actuel canal sans fil pour rendre une décision de l'exécution optimale. Semblable à Wang and Li 2004), l'approche en (Chen *et al.* 2004) manque de généralité et ne peut pas être appliqué pour diverses applications.

b) Le déchargement dans l'environnement dynamique

Cette sous-section introduit quelques approches pour traiter le déchargement dans un environnement de réseau dynamique (ex. changement d'état de la connexion et de la bande passante). Les modifications apportées à l'environnement peuvent causer des problèmes supplémentaires. Par exemple, les données transmises peuvent ne pas atteindre leurs destinations, ou les données exécutées sur le serveur seront perdues alors qu'elles doivent être renvoyées à l'expéditeur.

S. Ou et *al.* (Ou *et al.* 2007) analysent le rendement de systèmes de déchargement opérant dans des environnements sans fil. Dans ce travail, les auteurs tiennent compte de trois circonstances de l'exécution d'une application, ainsi estimé l'efficacité de déchargement. Ils sont les cas où l'application est exécutée localement (sans déchargement), l'application est exécutée dans les systèmes de déchargement idéal (sans échec), et l'application est exécutée avec la présence de déchargement et le recouvrement depuis la défaillance. Dans le dernier cas, lorsqu'une défaillance se produit, l'application sera re-déchargé. Cette approche ne se ré-décharge que les sous-tâches échouées, améliorant ainsi le temps d'exécution. Cependant, cette solution a quelques limitations. C'est-à-dire l'environnement mobile est considéré comme un réseau sans fil ad hoc (c.-à-d. la connectivité à large bande n'est pas prise en charge). De plus, au cours de l'exécution de déchargement, le débranchement d'un périphérique mobile est traité comme un échec.

M. H. Tang et *Cao.* (Tang and Cao 2006) considèrent trois changements de l'environnement commun figurant au tableau 2.2 et explique les solutions appropriées pour le déchargement dans les différents environnements. Par exemple, dans le cas de changement l'état de la connexion (ex. la déconnexion pendant l'exécution du programme), le serveur va vérifier périodiquement l'état de la connexion avec le client et maintient les informations de l'exécution concernant les tâches en cours d'exécution. Lorsque la déconnexion est récupérée, le serveur envoie les résultats de l'exécution pour le client. Si le serveur peut ne pas reconnecter au client, le serveur attend pour l'intervalle de temps prédéfini et les tâches seront supprimées.

Toutefois, l'inconvénient de ces méthodes est qu'elles ne sont que des solutions générales et qu'elles ne mentionnent pas une méthode détaillée pour aborder la question de la partition dynamique, c'est-à-dire, la manière de partitionner l'application.

Modifications	Niveau de priorité	Description
Niveau de puissance côté client	1	L'alimentation peut être divisée en niveaux de puissance insuffisante et suffisante, Ce qui dépendra de la situation particulière.
L'état de la connexion	2	L'état de la connexion peut être altéré, déconnectée de réseau mobile ou re- connecté au réseau mobile
Bande passante	3	La bande passante varie de temps à autre et dépend de plusieurs facteurs comme l'état de la circulation de réseau

Table 2. 2 Les modifications communes de l'environnement de mobile Computing (Tang and Cao 2006).

Deuxièmement, dans le choix de partitionnement de l'étape, le système va choisir une politique de cloisonnement approprié de sorte que la consommation totale d'énergie est réduite au minimum. Enfin, pour répondre à la question de la sécurité, les auteurs soulignent que les modules contenant des données sensibles seront exécutés localement. Les données sensibles sont marquées selon le programmeur par des annotations. Ce système tient compte à la fois du problème de sécurité et d'application de partitionnement. Toutefois, il manque de précision depuis la partition est basé sur un modèle de prédiction au moyen d'une analyse hors ligne.

2.6.2.2 La sécurité

La protection de la vie privée de l'utilisateur et données/application de l'adversaire est une clé pour établir et maintenir la confiance des consommateurs dans la plateforme mobile, en particulier dans les MCC. Dans ce qui suit, les questions relatives à la sécurité dans les MCC sont introduites en deux catégories : la sécurité pour les utilisateurs mobiles et la sécurité des données. En outre, certaines solutions pour répondre à ces questions sont examinées.

a) La sécurité des utilisateurs mobiles

Les appareils mobiles tels que les téléphones cellulaires, PDA et smartphones sont exposés à de nombreuses menaces de sécurité comme les codes malveillants (ex. virus,

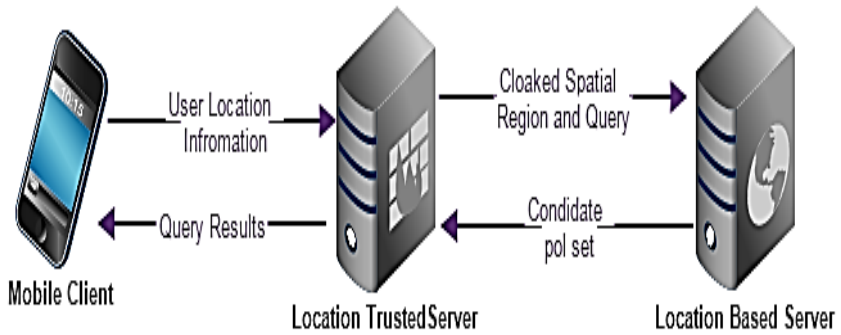
chevaux de Troie) et leur vulnérabilité. En plus, les appareils mobiles avec GPS intégré peuvent causer des problèmes de protection de la vie privée pour les abonnés. Deux principaux enjeux sont comme suit.

La sécurité des applications mobiles à base de Cloud : L'installation et l'exécution de logiciels de sécurité tels que Kaspersky, McAfee et programmes antivirus AVG sur les appareils mobiles sont les plus simples moyens pour détecter les menaces de sécurité (ex. virus, et codes malveillants) sur les périphériques mobiles. Toutefois, les périphériques mobiles sont limités dans leur alimentation et traitement, les protégeant de menaces est plus difficile que celle d'ingénieux dispositif (ex. PC), car il est impossible de continuer à fonctionner le logiciel de détection de virus sur les périphériques mobiles (Dinh *et al.* 2013).

Pour démontrer l'efficacité de l'utilisation de Cloud Computing afin de détecter les logiciels malveillants sur les appareils mobiles, (Oberheide *et al.* 2008) présente un paradigme dans lequel la détection d'attaque pour un smartphone est exécutée sur un serveur distant dans le Cloud. De même, au lieu d'exécuter un programme antivirus localement, le smartphone enregistre seulement un minimum de traces d'exécution, et les transmet au serveur de sécurité dans le Cloud. Ce paradigme non seulement améliore l'efficacité de la détection des applications malveillantes, mais améliore également la durée de vie de la batterie jusqu'à 30 %.

Protection de la vie privée : avec les avantages de dispositifs de positionnement GPS, le nombre des utilisateurs mobiles augmente à l'aide de *Location Based Services* (LBS). Toutefois, les LBS sont confrontées à un problème de confidentialité lorsque les utilisateurs mobiles fournissent des renseignements privés tels que leur emplacement actuel. Ce problème est encore plus grave si l'adversaire sait des informations importantes de l'utilisateur.

Location Trust Service (LTS) (Zhangwei and Mingjun 2010) est présentée pour aborder cette question. Comme illustré dans la figure 2.5, après avoir reçu les demandes des utilisateurs mobiles, LTS recueille les informations sur leur emplacement dans certaines zones et enveloppes les informations appelées « Cloakes Region » fondé sur « k-anonymat » concept (Sweeney 2002) afin de masqué l'information de l'utilisateur. Cloakes Region est envoyée au LBS, de sorte que le LBS sait que les informations générale



sur les utilisateurs, mais ne peut pas les identifier.

Figure 2. 5 Architecture globale de Spatial Cloaking (Sweeney 2002).

b) La sécurité des données sur les Clouds

Bien que les utilisateurs mobiles et les développeurs des applications bénéficient de stocker une grande quantité de données/applications sur un Cloud, ils devraient être prudents de traiter les données/applications en termes de leur intégrité, d'authentification et de droits numériques. Les problèmes liés aux données dans les MCC sont comme suit.

Intégrité: les utilisateurs mobiles inquiètent souvent sur l'intégrité des données dans le Cloud. Plusieurs solutions sont proposées pour remédier à ce problème (ex. (Andrew and van Steen 2007)). Toutefois, ces solutions ne prennent pas la consommation d'énergie des utilisateurs mobiles en compte. (Itani *et al.* 2010) considèrent la question de la consommation d'énergie. Ce régime est constitué de trois composants principaux : un client mobile, un service de stockage Cloud, et un tiers de confiance.

Authentification: (Chow et al. 2010) présentent une méthode d'authentification en utilisant le Cloud Computing pour sécuriser l'accès aux données dans les environnements mobiles approprié. Ce régime combine TrustCube (Song *et al.* 2009) et l'authentification implicite (Jakobsson *et al.* 2009), (Jakobsson *et al.* 2009)afin d'authentifier les clients mobiles.

TrustCube est une politique axée sur la plateforme d'authentification de normes ouvertes, et elle appuie sur l'intégration des différentes méthodes d'authentification. Les auteurs ont construit un système d'authentification implicite à l'aide de Mobile Data (ex. appel, messages SMS journaux sur le site web, et l'emplacement) pour environnement mobile existant. Le système nécessite une entrée contrainte qui rend difficile pour les utilisateurs mobiles d'utiliser des mots de passe complexes. En conséquence, ce qui conduit souvent à l'utilisation de mots de passe courts et simples ou des broches. Figure 2.6 montres l'architecture système et la manière dont le système garantit l'accès des utilisateurs mobiles.

Lorsqu'un serveur Web reçoit une requête d'un client mobile, le serveur Web redirige la demande vers le système de *service intégré authentifié* (IA) avec les détails de la demande. Le service IA récupère la politique pour la demande d'accès, extrait les informations qui doivent être recueillies et envoie une demande de renseignements à l'*IA Server* par l'intermédiaire de la TNC (*Trusted Network Connect*) protocole. Les serveurs IA reçoivent la demande, génèrent un rapport et le renvoient à l'IA Service. Après cela, l'IA Service applique la règle d'authentification dans la politique et détermine le résultat d'authentification (si oui ou non le client mobile est correctement authentifié pour la demande d'accès) et envoie le résultat d'authentification au serveur web. En se fondant sur le résultat d'authentification, le *Serveur Web* soit fourni le service ou rejette la demande.

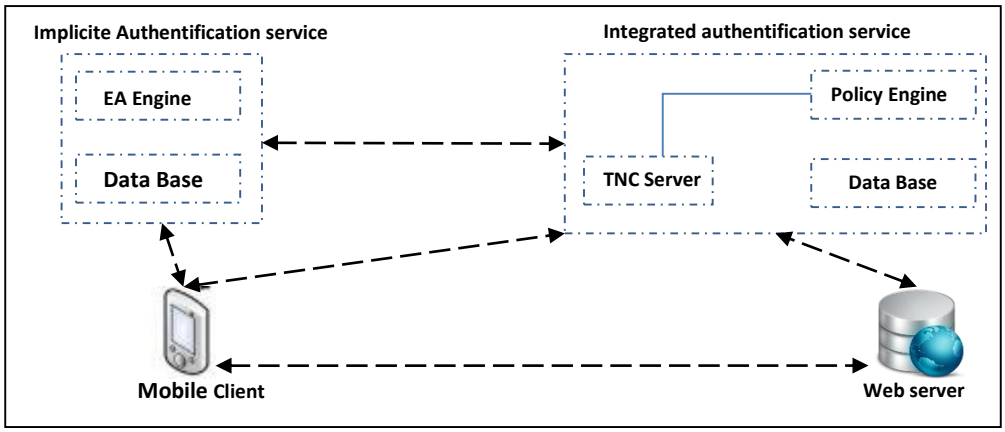


Figure 2. 6 Architecture TrustCube .

La gestion des droits numériques: le contenu numérique non structuré (ex. vidéo, photo, audio et e-book) a souvent été piraté et illégalement distribué. La protection de ces matières des accès illégaux est d'une importance cruciale pour les fournisseurs de contenu dans les MCC comme Cloud Computing et réseaux peer-to-peer.

E. Cuervo et *al.* (Cuervo *et al.* 2010) propose Phosphore, *Cloud based Mobile Degital Right Management* (DRM) régime avec une carte sim de téléphone mobile pour améliorer la flexibilité et réduire la vulnérabilité de sa sécurité à un faible coût. Les auteurs conçoivent un état de licence Word (Licence State Word) situé dans une carte SIM. Ce protocole est basé sur le protocole *Application Data Protocol Unit* (APDA). Lorsqu'un utilisateur mobile reçoit les données chiffrées (ex., flux vidéo) à partir du serveur via protocole RTP, il/elle utilise la clé de déchiffrement d'une carte SIM via commande UDPA à décoder. Si le décodage est réussi, l'utilisateur mobile peut regarder cette vidéo sur son téléphone. L'inconvénient de cette solution est qu'elle est toujours basée sur carte sim de téléphone mobile, de sorte qu'il ne peut s'appliquer pour d'autres types d'accès, c'est-à-dire, un ordinateur portable à l'aide du Wi-Fi pour accéder à ces matières.

2.6.2.3 Amélioration de l'efficacité de l'accès aux données

Avec un nombre croissant de services de Cloud Computing, la demande d'accès aux ressources de données (ex. image, de dossiers et de documents) sur le Cloud augmente. En conséquence, une méthode de traiter les données (c.-à-d. stocker, gérer et accéder à des ressources données sur les Clouds) devient un défi important. Toutefois, la manipulation de données sur les Clouds n'est pas un problème facile en raison de la faible bande passante, mobilité, et la limitation de la capacité des ressources de périphériques mobiles. Pour les fournisseurs de stockage Cloud commercial (ex. Amazon S3), toutes les opérations d'E/S sont prises par le fournisseur. Les opérations d'E/S sont exécutées à un niveau-fichier en général, de sorte que cela augmente le coût du réseau de communication et de service pour les utilisateurs mobiles.

Shen et *al.*(Shen et al. 2010) proposent un algorithme dans les opérations d'E/S qui sont exécutées à un bloc-niveau. L'algorithme utilise journal de transaction E/S structuré (Rosenblum and Ousterhout 1992) afin de minimiser le nombre de blocs d'E/S au niveau des opérations. L'idée principale ici est d'effectuer le stockage en Cloud avec le nombre optimal de blocs de données de façon adaptative en changeant avec E/S et la politique de

paiement de stockage. Les auteurs démontrent que, par l'expérimentation, la solution proposée réduit le total considérablement les coûts d'E/S jusqu'à 54 % par rapport à la gestion des données au niveau fichier dans Amazon S3. Toutefois, cette solution ne permet pas d'envisager sur les méthodes d'accès pour s'adapter à cette nouvelle gestion des données.

2.6.2.4 Conscience du contexte des services mobile à base de Cloud

Il est important pour les fournisseurs de services d'assurer la satisfaction des utilisateurs mobiles en réalisant leurs exigences et offrir des services appropriés à chacun des utilisateurs. Beaucoup de travaux de recherche tentent d'utiliser des contextes locaux (ex. types de données, état du réseau, les environnements de périphérique, et des préférences de l'utilisateur) pour améliorer la qualité de service (QoS).

F.A Samimi et *al.* (Samimi *et al.* 2006) proposent un modèle, appelé *Mobile Cloud Services* (MSCs), qui est une extension du de *Clouds Service Paradigm* (McKinley *et al.* 2006). Dans ce modèle, lorsqu'un client utilise un service Cloud, la demande de l'utilisateur tout d'abord va à une passerelle de service. La passerelle permettra de choisir un proxy principal pour répondre aux exigences (ex. le chemin le plus court et le temps aller-retour minimum) puis envoie le résultat à l'utilisateur. Dans le cas d'une déconnexion, MSCs établira les proxys transitoires (Samimi *et al.* 2004) pour les appareils mobiles afin de surveiller le chemin de service, et d'appuyer la reconfiguration dynamique (avec un minimum d'interruption). Les avantages de ce modèle sont que le modèle répond à la question de déconnexion et peut maintenir la qualité de service à un niveau acceptable.

P. Papakos et *al*. ont développé un middleware, appelé VOLARE (Papakos *et al.* 2010) , intégré dans l'appareil mobile, qui contrôle les ressources et contextes de l'appareil mobile, ainsi ajusté dynamiquement les exigences des utilisateurs au moment de l'exécution. Comme illustré dans la figure 2.7, lorsqu'un utilisateur lance une application mobile sur son appareil qui requiert des services sur le Cloud, cette demande est transitée au *Mobile OS* avant qu'il ne soit envoyé au *Service Request Module*. Dans le même temps, *Mobile OS* envoie simultanément les données de contexte à *Context Monitoring Module*, et les données de surveillance de la qualité de service à *QoS Monitoring Module*. Adaptation Module recevra la demande de service à partir de Service *Request Module* et traite cette demande avec l'alerte reçue du *Context Monitoring Service* s'il y a des différences de contextes et de notifications sur la qualité de service de *Quality Of Service Monitoring Module* lors de l'exécution.

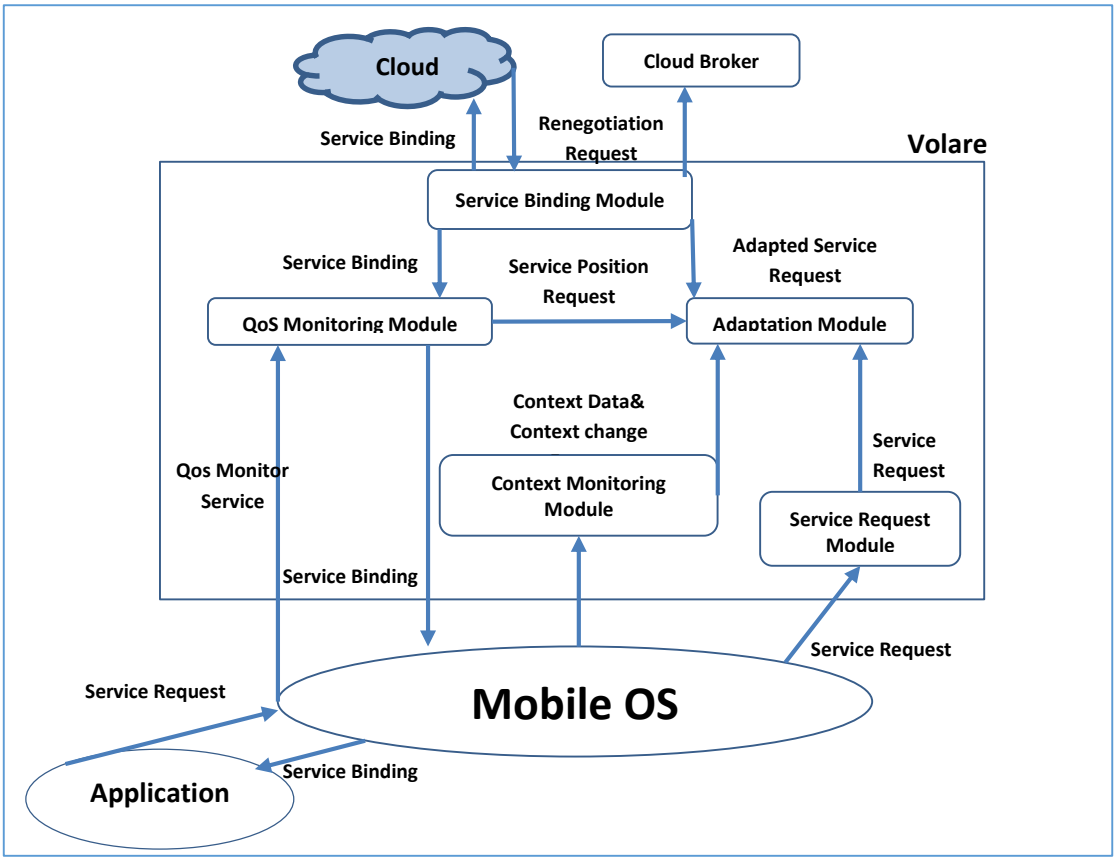


Figure 2. 7 L'architecture de Volare .

Par conséquent, *Adaptation Module* peut offrir le service approprié de demandes fondées sur le contexte et les données de ressource. Comme *Quality Of Service Monitoring Module* s'exécute périodiquement et vérifie si le niveau de QoS fourni par les fournisseurs est inférieur à un niveau acceptable de la demande, *Service Request Module* sera notifié pour lancer une nouvelle demande de découverte d'un nouveau service satisfaisant aux nouvelles exigences. L'avantage de ce modèle est qu'il peut reconnaître automatiquement les modifications dans les contextes sur le périphérique mobile via la fonctionnalité dépendante de modules, ainsi offrant un service approprié à l'utilisateur .

2.7 Discussion

D'après l'étude établit dans ce chapitre sur le Mobile Cloud Computing, nous discutons que due au fait que ce paradigme et encore récent reste plusieurs problématiques à aborder différentes à celles émotionnées dans les sections précédentes. Parmi les questions que nous attestons sont importante a posés sont :

Avec l'intégration du Cloud au Mobile Computing, les Applications mobiles à base de Cloud deviennent plus compliquées et les développer de façon aléatoire n'est pas une méthode efficace afin de produire des applications performantes. Alors, pourquoi ne pas modéliser ces applications avant leurs déploiements dans un environnement réel ?

Le Cloud offre une grande variété de services suivant une métrique de paiement. Par conséquent, le développeur doit faire face à deux problèmes très importants. Non seulement la phase de vérification de l'application pour gérer les bugs et les pannes est

couteuse, mais aussi le cas de modification de service Cloud (ex. la suppression du service, ou l'augmentation du coût) avec lequel l'application mobile interagit. Dans ce cas, pourquoi ne pas simuler le comportement de l'application dans un environnement Cloud simulé avant le déploiement dans un enivrement Cloud réel et payant ?

L'infrastructure Cloud offres une très grande variété de service qui différés en termes de Qualité et le prix. Nous pensons qu'une modification dynamique des services et indispensable pour garantir la satisfaction des utilisateurs, mais comment le faire ? Reste une question sans réponse.

2.8 Conclusion

Le Mobile Cloud Computing est l'une des tendances de la technologie mobile dans le futur, puisqu'il combine les avantages des deux systèmes informatiques mobiles et le Cloud Computing, fournissant ainsi des services optimaux pour les utilisateurs mobiles. Selon une étude récente par ABI Research, une entreprise établie à New York, plus de 240 millions d'entreprises utiliseront les services Cloud au moyen d'appareils mobiles d'ici 2015. Par conséquent, les revues du mobile Cloud Computing seront augmentées à 5,2 milliards de dollars. Avec cette importance, ce chapitre fourni un aperçu sur le Mobile Cloud Computing, ses définitions, architecture, et les avantages ont été présentés. Les applications mobiles à base de Cloud, notamment m-Commerce, m-Learning, et m-Healthcare ont été discuté ce qui montre clairement l'applicabilité du MCC pour une large gamme de services mobiles. Ensuite, les questions et les approches relatives à MCC ont été données.

Chapitre 3

Modélisation et ingénierie dirigée par les modèles

3.1 Introduction

L'ingénierie des logiciels fournit sans cesse des nouvelles technologies qui facilitent la mise en œuvre des systèmes tout en améliorant leurs qualités. Le revers de la médaille de ces technologies est les plateformes d'exécution, due au fait qu'elles complexifient énormément les systèmes informatiques. Cela place les entreprises dans une situation inconfortable, car elles devront choisir entre adopter une nouvelle plate-forme et subir le coût de la migration ou ne pas l'adopter et prendre le risque de voir la compétition devenir plus difficile grâce au choix inverse des concurrents.

La complexité des plates-formes d'exécution, conçues afin de faciliter le développement et la maintenance des systèmes, mais elles sont considérée comme un inconvénient majeur et un frein à l'évolution. Pour y faire face, il était nécessaire de définir une approche permettant de gérer la complexité. Cette approche se devait d'être flexible et générique afin de pouvoir s'adapter à tout type de plate-forme.

L'ingénierie logicielle guidée par les modèles, ou Ingénierie Dirigée par les Modèles (IDM), correspond à la définition d'une telle approche. IDM favorise la séparation des préoccupations entre la logique métier des systèmes informatiques et les plates-formes utilisées et se base principalement sur l'utilisation des modèles. Un cas particulier de l'IDM est le *Model Driven Architecture* (MDA) qui nous détaillons plus loin dans ce chapitre.

Nous présentons dans ce chapitre les principes clés de cette ingénierie. Nous introduisons dans un premier temps la notion de l'IDM (cf. section 3.2) et après la notion de l'MDA (cf. section 3.3) vue comme variante particulière d'IDM enfin on termine par la modélisation des applications mobiles (cf. section 3.4).

3.2 IDM : l'ingénierie dirigée par les modèles

L'ingénierie dirigée par les modèles est une des approches modernes du développement des logiciels. 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 et le principe du «*tout est modèle* ». Se pose alors la question des concepts essentiels. L'approche orientée objet est basée sur deux relations fondamentales: la relation « *Instance De »*qui permet d'introduire la notion de classe et la relation « *Hérite*

*De »*qui permet d'introduire la notion de superclasse. Il s'agit de déterminer quels sont les relations et les concepts essentiels de l'IDM. Bien qu'aucune réponse à cette question ne puisse être définitive à ce stade des recherches, il apparaît de plus en plus consensuel que deux relations sont fondamentales. La première relation, appelée « *Représentation de »* est liée à la notion de *modèle*, alors que la relation « *Est conforme à »* permet de définir la notion de modèle par rapport à celle de *méta modèle*.

3.2.1 Définition de l'IDM

Pour définir l'IDM, nous choisissons la définition de Mellor et *al*. Qui est suffisamment large tout en étant claire sur l'intuition qui la fonde. L'IDM est simplement la notion que nous pouvons construire un modèle d'un système puis le transformer en la chose réelle (Mellor *et al.* 2003).

3.2.2 Pourquoi l'ingénierie des modèles (*IDM*)

Le modèle est devenu le paradigme principal par lequel l'industrie du logiciel pourra améliorer l'automatisation du développement, des organismes tels que l'OMG et les chercheurs en génie logiciel cherchent à enrichir les méta modèles qui sont utilisés dans la conception des applications ou à en définir de nouveau, à rendre la création de nouveaux espaces de modélisation plus facile et plus adaptée aux besoins des utilisateurs et à automatiser les différentes étapes de modélisation nécessaires à l'élaboration d'un logiciel fini. Ainsi, pour produire un logiciel qui répond aux besoins des utilisateurs, il est nécessaire de fournir un passage des modèles d'un niveau d'abstraction à un autre ou d'un espace technologique à un autre.

L'id est peut être considéré comme une ingénierie générative, par laquelle tout ou partie d'une application informatique est engendrée à partir de modèles. Dans cette nouvelle perspective, les modèles se placent en premier plan parmi les artéfacts de développement des systèmes, mais doivent en contrepartie être suffisamment précis et riches afin de pouvoir être interprétés ou transformés par des machines. Le procédé de développement des systèmes est une forme d'une séquence de transformations de modèles demi-ordonnée. Chaque étape de transformation prend en entrée un ou des modèles et produit en sortie un ou des modèles jusqu'à l'obtention d'artéfacts exécutables (Diaw *et al.* 2010).

Au sens large de terme, L'IDM est la discipline qui place les modèles au centre des processus d'ingénierie logicielle. Avec cette approche, le code final exécutable n'est plus considéré comme l'élément central dans le processus de développement, mais comme un élément important qui est le résultat d'une transformation de modèles.

Pour mieux expliquer la logique et l'apport de l'IDM, nous présentons dans la prochaine sous-section section une approche principale de développement logiciel basée sur les modèles qui contribuent fortement à l'émergence de l'IDM : Model Driven Architecture (MDA).

3.3 L'approche MDA

Dans l'ingénierie dirigée par les modèles, les modèles sont au cœur de la conception et le développement des systèmes. Par conséquent, l'IDM est l'avenir de l'industrie des logiciels et comme dit Bill Gates : « Modéliser est le futur, et je pense que les sociétés qui travaillent dans ce domaine ont raison » (Blanc 2005).

3.3.1 Définition de MDA

MDA est défini par OMG en 2000 comme une variante particulière de l'ingénierie dirigée par les modèles. L'approche MDA favorise l'utilisation massive des modèles et répond aux comment, quand, quoi et pourquoi modéliser. En répertoriant toutes les bonnes pratiques, elle vise à mettre en premier plan les qualités favorables des modèles, comme la pérennité, la productivité et la notion de multiplateformes d'exécution. MDA comporte la définition des standards, notamment UML, MOF et XMI.

3.3.2 Principe et objectif

Le MDA s'appuie fortement sur le standard UML pour décrire indépendamment des modèles correspondant à des différentes phases du cycle de développement d'un logiciel. En outre, le MDA recommande l'élaboration de modèles:

- D'exigence (Computation Independent Model – CIM) ;
- D'analyse et de conception (Platform Independent Model – PIM) ;
- De code (Platform Specific Model – PSM) (Combemale 2008).
- Le MDA fournit un processus et met en œuvre des outils pour :
- Spécifier un système indépendamment de la plate-forme qui le supporte, et donc réaliser un PIM ;
- Enrichir ce modèle par des étapes successives ;
- Spécifier les plates-formes ;
- Choisir une plate-forme particulière pour le système ;
- Transformer la spécification du système (PIM) en une autre spécification pour une plate-forme particulière (PSM) ;
- Raffiner le PSM jusqu'à obtenir une implémentation exécutable.

Les trois objectifs préliminaires de MDA sont *la portabilité, l'interopérabilité et la réutilisabilité* à travers une architecture de séparation des préoccupations. L'objectif majeur de MDA est l'élaboration de modèles pérennes, indépendants des détails de mise en œuvre (J2EE, .Net, PHP ou autres), afin de permettre la génération automatique du code des logiciels et d'obtenir un gain significatif de productivité(Combemale 2008).

3.3.3 Architecture de l' MDA

Dans la figure 3.1, l'architecture MDA se découpe en quatre couches. Dans la première couche se trouve le standard UML (*Unified Modeling Language*), MOF (*Meta_Object Facility*) et CWM (*Common Warehouse Metamodel*). Dans la couche suivante, se trouve aussi un standard XMI (*XML Metadata Interchange*) qui permet le dialogue entre les middlewares (java, corba, .net et web services). La troisième couche contient les services qui permettent de gérer les évènements, la sécurité, les répertoires et les transactions. Enfin, la dernière couche propose des Frameworks adaptables à différents types d'applications à savoir finance, télécommunication, transport, espace, médecine, commerce électronique et manufacture…).

3.3.3.1 Standards de l'OMG

L'OMG est un consortium regroupant des industriels et des chercheurs dont l'objectif principal est d'établir des standards pour résoudre, entre autres, les problèmes d'interopérabilité entre les systèmes d'information. Ces standards définissent l'infrastructure du MDA et qui sont :

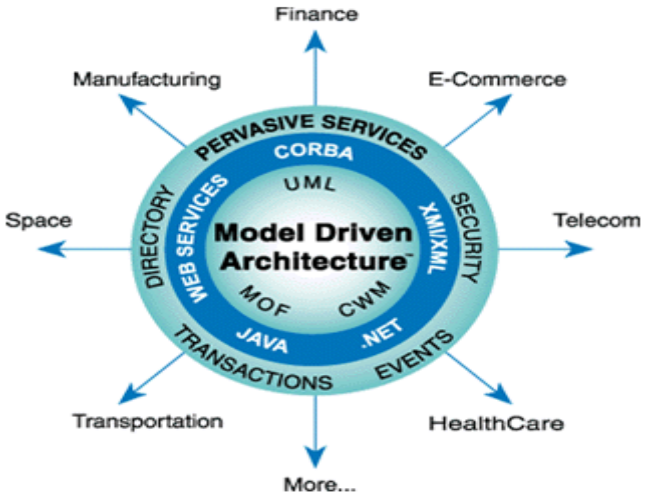


Figure 3. 1 L'architecture de l'MDA

a) Unified Modeling Language (*UML*)

UML est le standard de modélisation des logiciels à objets. UML permet la modélisation d'architectures, de structure d'objets, d'interactions entre ces objets, de données, de composants

Le MDA préconise fortement l'utilisation d'UML pour l'élaboration de PIMs et de la plupart des PSMs : les spécificités de chaque plate-forme peuvent être modélisées grâce aux mécanismes d'extension d'UML (stéréotypes, valeurs marquées, contraintes). En fait, on peut définir pour chaque système un profil qui regroupe les éléments nécessaires à ses caractéristiques. Si nous considérons qu'UML est un langage, les profils sont alors des dialectes de ce langage et ces dialectes sont fortement utilisés dans le MDA. Par exemple, le profil EJB permet l'élaboration de PSM pour la plate-forme EJB.

À l'heure actuelle, UML est dans sa version *2.x*. Cette version contient entre autres une évolution du langage d'expression de contrainte OCL. Sont aussi redéfinis le noyau d'UML (infrastructure) et les extensions (superstructure) ; en particulier, les profils UML. Un profil UML offre une extension à UML pour inclure certaines caractéristiques supplémentaires spécifiques à un domaine particulier .

b) Meta Object Facility (*MOF*)

Le langage MOF fournit le standard de métamodélisation et d'échange de constructions utilisées par MDA. Les autres standards de l'OMG, comme UML et CWM, sont définis par des constructions MOF, ce qui permet de les relier entre eux. Le MOF est un méta méta-modèle, ou un modèle du méta-modèle. Il définit les éléments essentiels, la syntaxe et la structure des méta-modèles utilisés pour construire des modèles orientés objet ; grâce à des diagrammes ressemblants étrangement à des diagrammes de classe UML. Dans ces diagrammes, les classes (que l'on appelle des méta-classes) représentent les concepts à définir et les associations (que l'on appelle des méta-associations) représentent les relations entre ces concepts. Les méta-classes et les méta-associations sont contenues dans des packages .

L'intérêt du MOF est qu'il permet de faire interopérer des méta-modèles différents. Une application MOF peut manipuler un modèle à l'aide d'opérations génériques sans connaissance du domaine. L'exemple le plus significatif des opérations génériques est l'échange de modèles, par l'utilisation de XMI, quels que soient leurs méta-modèles (à condition qu'ils soient conformes à MOF).

c) Xml Metadata Interchange(*MLX*)

XMI est le standard de l'OMG qui fait la jonction entre le monde des modèles et le monde XML de W3C (World Wide Web Consortium). Le standard XMI permet de décrire une instance du MOF sous forme textuelle, grâce au langage XML en définissant la manière d'utilisation de ses balises. Il permet la génération de DTDs (Document Type Définitions) et de schémas XML à partir de méta-modèles MOF. Les modèles sont alors traduits dans des documents XML conformes à leurs DTD correspondantes. XMI offre ainsi une solution pour représenter des objets et leurs associations sous forme textuelle. De plus, puisque XMI est basé sur XML, les méta-données (tags) et les instances (éléments) sont regroupées dans le même document, ce qui permet à une application de comprendre les instances grâce à leurs méta-données. Le XMI est ainsi le format d'échange standard entre les différents outils MDA.

d) Common Warehouse Meta model (*CWM*)

CWM est le standard de l'OMG qui traite des entrepôts de données. Il couvre le cycle de vie complet de modélisation, construction et gestion des entrepôts de données. L'approche préconisée par ce standard pour la migration entre systèmes hétérogènes est une approche MDA (la création de modèles et leurs transformations).CWM définit un méta-modèle qui représente les métadonnées aussi bien métiers que techniques qui sont le plus souvent retrouvées dans les entrepôts de données.

CWM définit actuellement les méta-modèles des principaux types d'entrepôts de données (Relationnel, Objet, XML,....) et propose des règles de transformation entre ceux-ci. Le CWM comprend un certain nombre de méta-modèles concernant la représentation des données, l'analyse et la gestion des entrepôts de données. Les méta-modèles de données permettent de modéliser des ressources comme les bases de données relationnelles et les bases de données orientées objet (Combemale 2008).

3.3.4 Modélisation MDA

Les standards cités précédemment sont centrés sur la notion de méta modèle qui décrit la structure des modèles et sur celle de méta-méta modèle. Nous donnons les définitions de : modèles, méta-modèles et Méta méta-modèle dans les paragraphes suivants. Ces trois notions sont schématisées dans la figure 3.2 ci-après.

3.3.4.1 Définition d'un modèle

Les modèles offrent de nombreux avantages. Ceux qui pratiquent UML ou tout autre langage de modélisation les connaissent bien. L'avantage le plus important qu'ils procurent est de spécifier différents niveaux d'abstraction, facilitant la gestion de la complexité inhérente aux applications. Les modèles très abstraits sont utilisés pour présenter l'architecture générale d'une application ou sa place dans une organisation, tandis que les modèles très concrets permettent de spécifier précisément des protocoles de communication réseau ou des algorithmes de synchronisation. Même si les modèles se situent à des niveaux d'abstraction différents, il est possible d'exprimer des relations de raffinement entre eux. Véritables liens de traçabilité, ces relations sont garantes de la cohérence d'un ensemble de modèles représentant une même application (Combemale 2008).

En informatique « *Un modèle a pour objectif de structurer les données, les traitements, et les flux d'informations entre entités*». On peut définir aussi un modèle comme une abstraction de la réalité, une image simplifiée d'un système donné ou bien une simple représentation d'un système.

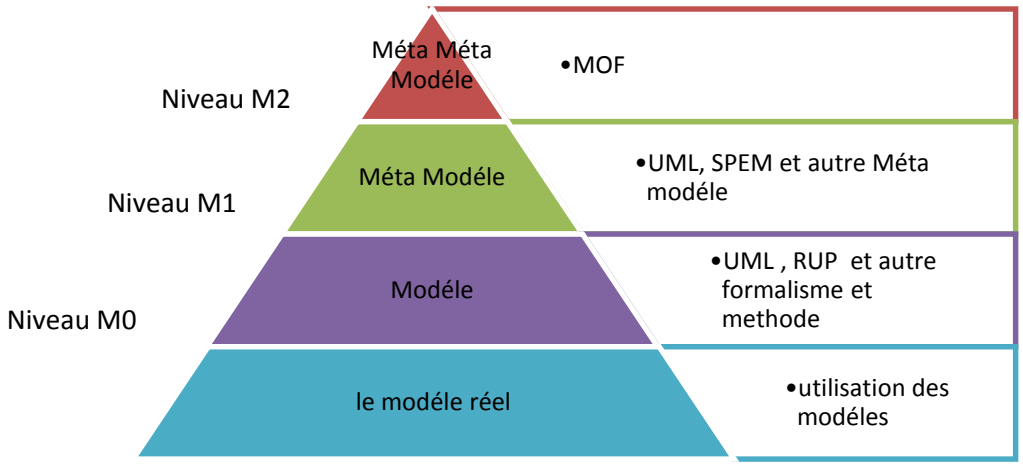


Figure 3. 2 Pyramide de modélisation de l'OMG

3.3.4.2 Les Objectifs et les avantages d'un modèle

La diversité des possibilités de modélisation ainsi que la possibilité d'exprimer des liens de traçabilité sont des atouts décisifs pour gérer la complexité. Un autre avantage incontestable des modèles est qu'ils peuvent être présentés sous format graphique, facilitant d'autant la communication entre les acteurs des projets informatiques.

Les modèles graphiques parmi les plus utilisés sont les modèles relationnels, qui permettent de spécifier la structure des bases de données. La représentation graphique de ces modèles offre un gain significatif de productivité.

Les mauvaises langues prétendent que modélisée est la meilleure façon de perdre du temps puisque, in fine, il faut de toute façon écrire du code. De même, au fameux dicton énonçant qu'un bon schéma vaut mieux qu'un long discours, on entend parfois répliquer qu'à un schéma peuvent correspondre plus de mille discours, selon la façon dont on l'interprète.

Ces critiques visent juste en cas d'absence de maîtrise des bonnes pratiques de modélisation, c'est-à-dire de l'ingénierie des modèles. C'est pourquoi il est essentiel d'acquérir de bonnes pratiques de modélisation afin de déterminer comment, quand, quoi et pourquoi modéliser et d'exploiter pleinement les avantages des modèles. En résument les bienfaits de la modélisation par les points suivants :

- La modélisation aide à comprendre un problème complexe ou une solution ;
- Communiquer à propos d'un problème complexe ou d'une solution ;
- Simuler le fonctionnement d'un système ;
- Abstrait: elle fait ressortir les points importants tout en enlevant les détails non nécessaires ;
- Compréhensible: elle permet d'exprimer une chose complexe dans une forme plus facilement compréhensible par l'observateur ;
- Précis: elle représente fidèlement le système modélisé ;
- Prédictif: elle permet de faire des prévisions correctes sur le système modélisé;
- Peu coûteux: elle est bien moins coûteuse à construire et étudier que le système lui-même.

3.3.4.3 Définition d'un Méta-modèle et d'un Méta méta-modèle

Un modèle est une abstraction et une simplification d'un système qu'il représente. Il offre donc une vision schématique d'un certain nombre d'éléments que l'on décrit sous la forme d'un ensemble de fait construit dans une intention particulière. Un modèle doit pouvoir être utilisé pour répondre à des questions que l'on se pose sur lui.

Il faut noter qu'il reste toutefois difficile de répondre à la question « Qu'est-ce que un bon modèle? ». Néanmoins, un modèle doit être suffisant et nécessaire pour permettre de

répondre à certaines questions du système qu'il représente, exactement de la même façon que le système aurait répondu lui-même. Le modèle doit se substituer au système pour permettre d'analyser de manière plus abstraite certaines de ses propriétés (Combemale 2008).

La notion de modèle dans l'IDM fait explicitement référence à la notion de *formalisme* ou de *langage de modélisation* bien défini. Un langage de modélisation est défini par une syntaxe abstraite, une syntaxe concrète et une sémantique. La syntaxe abstraite définit les concepts de base du langage. La syntaxe concrète définit le type de notation qui sera utilisé pour chaque concept abstrait qui peut être graphique, textuel ou mixte. Enfin, la sémantique définit comment les concepts du langage doivent être interprétés par les concepteurs, mais surtout par les machines (Combemale 2008).

Un modèle doit pouvoir être manipulé par une machine afin qu'il soit productif. Le langage de description de ce modèle doit être clair. Alors, la définition d'un langage de modélisation en prenant la forme d'un modèle est appelée *Méta-modèle*.

Un *méta-modèle* est un modèle permettant la définition d'un langage de description d'un modèle. Autrement dit, le méta-modèle représente (modélise) les éléments d'un langage, leurs relations ainsi que leurs contraintes, c'est-à-dire une spécification de la syntaxe du langage.

Dans MDA, il n'existe qu'un seul méta-formalisme, le MOF (Meta Object Facility). Aussi appelé Méta-méta modèle, le MOF permet de décrire des formalismes de modélisation, ou méta-modèles, permettant eux-mêmes d'exprimer des modèles. Il est méta-circulaire (c.-à-d. il peut se définir lui-même).

Le méta-modèle à son tour est exprimé dans un langage de méta-modélisation spécifié par le *Méta-méta-modèle*. Le langage utilisé au niveau du *Méta-méta-modèle* doit être suffisamment puissant pour spécifier sa propre syntaxe abstraite et ce niveau d'abstraction demeure largement suffisant (*méta-circulaire*). Chaque élément du modèle est une *instance* d'un élément du méta-modèle.

3.3.4.4 Méta-modèles et typage des modèles

Dans la figure3.3 Le niveau M0, contient les applications informatiques à modéliser, le niveau M1, contenant les différents modèles de l'application, le niveau M2, contenant les différents méta-modèles qui ont été utilisés, et le niveau M3, contenant le *Méta-méta-modèle* qui a permis de définir uniformément les méta-modèles.

Les relations existantes entre les niveaux M1-M2 et M2-M3 sont équivalentes. M2 définit la structure de M1 tout comme M3 définit la structure de M2. Rappelons que le MOF modèle définit sa propre structure.

Un méta-modèle définissait la structure d'un ensemble de modèles. Un ensemble de méta-modèles peut donc être vu comme un système de typage des modèles. En prenant pour socle un ensemble de méta-modèles, nous pouvons typer les modèles selon leur méta-modèle. Donc L'OMG définit 4 niveaux de modélisation.

- M0 : Système réel, système modélisé
- M1 : Modèle du système réel défini dans un certain langage
- M2 : Méta-modèle définissant ce langage
- M3 : Méta-méta-modèle définissant le méta-modèle(le MOF)

3.3.4.5 Relation entre systèmes, modèle et méta-modèle

Un modèle est dit *conforme* à un méta-modèle et constitue une *représentation* d'un système existant ou imaginaire. La relation entre un méta-méta-modèle et un méta-modèle est analogue à la relation entre un méta-modèle et un modèle. Cette relation est illustrée dans la (cf. figure 3.3)

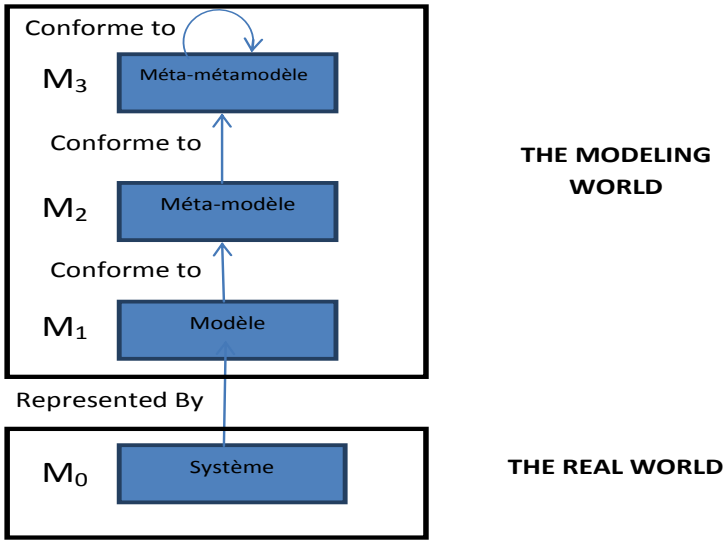


Figure 3. 3 Relation entre systèmes, modèle et méta-modèle

3.3.5 Application du modèle MDA dans le développement du logiciel

L'idée fondamentale de l'MDA est que les fonctionnalités du système à développer sont définies dans un modèle indépendant de la plate-forme (PIM : Platform Independent Model). Le PIM est ensuite traduit dans un ou plusieurs modèles spécifiques à la plate-forme (PSM : *Platform Specific Model*). Enfin, le PSM est utilisé pour générer le code source de la plate-forme ciblée (Blanc 2005). On va voir une définition détaillée pour chacun de ces modèles dans la section suivante.

3.3.5.1 Les modèles dans MDA

Le MDA comporte plusieurs modèles, « descriptions abstraites d'une entité du monde réel utilisant un formalisme donné » qui vont servir dans un premier temps à modéliser l'application, puis par transformations successives à générer du code (cf. figure 3.4). Aujourd'hui, la frontière entre les différents modèles n'est pas encore bien explicitée ni formalisée. Néanmoins, il est possible de donner une description de chacun d'eux.

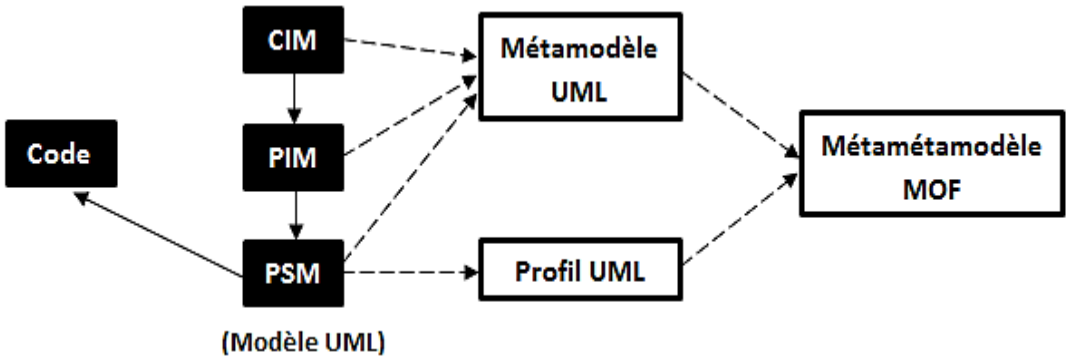


Figure 3. 4 Les trois types de modèles dans MDA (Laine 2013).

a) Le CIM (Computation Independent Model)

Selon OMG, le CIM est indépendant de tout système informatique. C'est le modèle métier ou le modèle du domaine d'application. Le CIM permet d'avoir une vue sur le système dans l'environnement où il opèrera, mais sans rentrer dans le détail de la structure du système ni de son implémentation. Il donne une représentation sur le code métiers du système. Il est utile, non seulement comme aide pour comprendre un problème, mais également comme source de vocabulaire partagé avec d'autres modèles. L'indépendance technique de ce modèle lui permet de garder tout son intérêt au cours du temps et il est modifié uniquement si les connaissances ou les besoins métier changent. Le savoir-faire est recentré sur la spécification CIM au lieu de la technologie d'implémentation. Dans les constructions des PIM (Platform Independent Model) et des PSM (Platform Spécifique Model), il est possible de suivre les exigences modélisées du CIM qui décrivent la situation dans lequel le système est utilisé, et réciproquement.

Le rôle des modèles d'exigences dans une approche MDA est d'être les premiers modèles pérennes. Une fois les exigences modélisées, elles sont censées fournir une base contractuelle variant, car validée par le client de l'application. Grâce aux liens de traçabilité avec les autres modèles, un lien peut être créé depuis les exigences vers le code de l'application.

UML et CIM : Pour faire un programme complexe, on a besoin de le modéliser (ex. par les diagrammes UML), afin d'explorer les solutions, les valider et pour montrer au client ce que sera l'application. L'élaboration des CIM avec UML fait débat. Rappelons qu'un CIM est un modèle d'exigences d'une application. Nous pouvons dire que les diagrammes de cas d'utilisation UML peuvent être considérés comme des CIM. Ces diagrammes permettent en effet de décrire les fonctionnalités qu'offre une application à son environnement. Il existe cependant d'autres approches d'expression des besoins, telles que celles supportées par Rational Requisite Pro. Voilà pourquoi MDA ne fait aucune préconisation quant à l'utilisation d'UML ou non pour exprimer les CIM (Combemale 2008).

b) Le PIM (Platform Independent Model)

Selon OMG, le PIM est indépendant de toute plate-forme technique (EJB, CORBA, .NET…) et ne contient pas d'informations sur les technologies qui seront utilisées pour déployer l'application. C'est un modèle informatique qui représente une vue partielle d'un CIM. Le PIM représente la logique métier spécifique au système ou le modèle de conception. Il représente le fonctionnement des entités et des services. Il doit être pérenne et durer au cours du temps. Il décrit le système, mais ne montre pas les détails de son utilisation sur la plate-forme. À ce niveau, le formalisme utilisé pour exprimer un PIM est un diagramme de classes en UML qui peut-être couplé avec un langage de contrainte comme OCL (*Object Constraint Language*). Il existe plusieurs niveaux de PIM. Le PIM peut contenir des informations sur la persistance, les transactions, la sécurité… Ces concepts permettent de transformer plus précisément le modèle PIM vers le modèle PSM.

c) Le PSM (Platform Specific Model)

Selon OMG, le PSM est dépendant de la plate-forme technique spécifiée par l'architecte. Le PSM sert essentiellement de base à la génération de code exécutable vers la ou les plates-formes techniques. Le PSM décrit comment le système utilisera cette ou ces plates-formes. Il existe plusieurs niveaux de PSM. Le premier, issu de la transformation d'un PIM, se représente par un schéma UML spécifique à une plateforme. Les autres PSM sont obtenus par transformations successives jusqu'à l'obtention du code dans un langage spécifique (Java, C++, C#, etc.). Un PSM d'implémentation contiendra par exemple des informations comme le code du programme, les types pour l'implémentation, les programmes liés, les descripteurs de déploiement.

Le rôle des modèles de code est principalement de faciliter la génération du code. Ils sont donc essentiellement productifs, mais ne sont pas forcément pérennes. L'autre caractéristique importante des modèles de code est qu'ils font le lien avec les plates-formes d'exécution. Cette notion de plate-forme d'exécution est très importante dans MDA car c'est elle qui délimite la fameuse séparation des préoccupations.

Il est parfois difficile de différencier le code des applications des modèles de code. Pour MDA, le code d'une application se résume à une suite de lignes textuelles, comme un fichier Java, alors qu'un modèle de code est plutôt une représentation structurée incluant, par exemple, les concepts de boucle, condition, instruction, composant, événement, etc. L'écriture de code à partir d'un modèle de code est donc une opération assez triviale.

3.3.6 Processus de développement général de MDA

Ce processus est proposé par Mike Rosen (cf. figure 3.5). Pour modéliser un système, en utilisant des outils de modélisation comme: Rational Rose, ArgoUML, Poisedon, Magic Draw,… En réalité, on commence le cycle de développement par modeler un modèle métier ou un CIM. Ce modèle est le résultat de la phase analyse. À partir du CIM, on ajoute des informations nécessaires pour obtenir un PIM. Puis on fait la transformation de modèle pour obtenir le PSM depuis un PIM. Le processus de transformation de modèle

peut être réalisé automatiquement ou manuellement. Ça dépend de l'outil MDA que l'on est en train d'utiliser. Enfin, depuis un PSM, on peut faire la génération de code source afin d'obtenir le code qui est prêt à être déployé.

La génération de code à partir des PSM n'est quant à elle pas considérée comme une transformation de modèle à part entière. MDA envisage aussi les transformations inverses : code vers PSM, PSM vers PIM et PIM vers CIM.

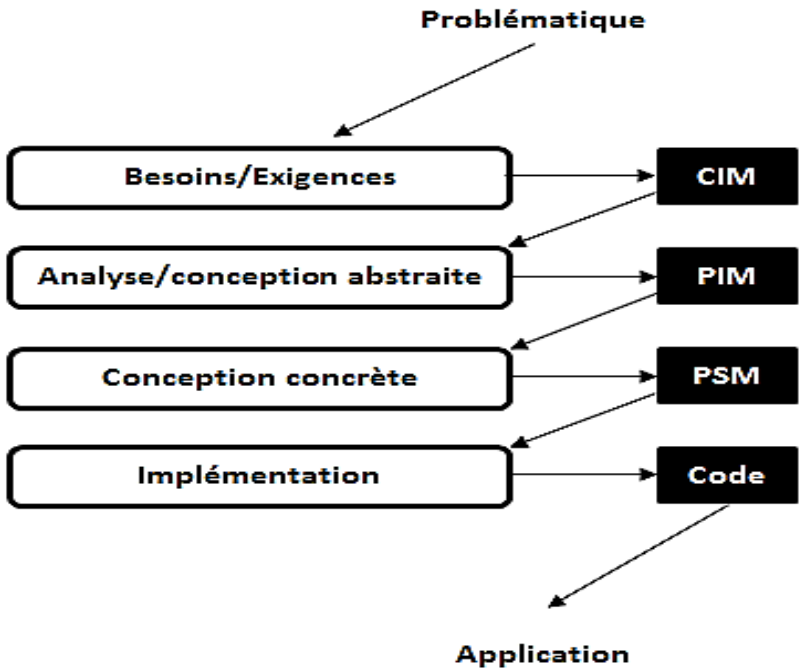


Figure 3. 5 Le processus de développement de MDA (Laine 2013).

MDA représente à l'aide de modèles toutes les informations permettant la construction d'applications informatiques. La mise en œuvre de MDA induit donc nécessairement la création d'un nombre important de modèles. Chacun de ces modèles contient une partie de l'information utile pour la génération des applications informatiques.

La mise en production de MDA passe par la mise en relation de ces modèles et plus précisément par leurs transformations respectives. C'est grâce aux transformations de modèles qu'il est possible, par exemple, d'obtenir une traçabilité entre des modèles très abstraits, proches des besoins exprimés par les utilisateurs, et des modèles très concrets, proches des plates-formes d'exécution.

3.3.7 Avantage de la démarche MDA

L'approche MDA promet de nombreux avantages, notamment :

- L'amélioration de la portabilité due à la séparation des connaissances métiers de la technologie spécifique d'implémentation ;
- L'augmentation de la productivité due à la transformation automatique ;

- L'amélioration de la qualité due à la réutilisation de patrons bien éprouvés avec les bonnes pratiques de transformation ;
- L'amélioration de la maintenabilité due à une bonne séparation des préoccupations et une meilleure cohérence et traçabilité entre les modèles et le code.

Il en ressort que le couple (Qualité-Délais), qui demeure la raison d'être du génie logiciel est garantie par l'approche MDA et ce, en s'appuyant sur son idée de base et son mode opératoire, à savoir la séparation, donc indépendance, des préoccupations métiers et technologie des plateformes et par les transformations automatiques de modèles (Czarnecki and Helsen 2003).

3.4 Modélisation des applications mobiles

La popularité des plateformes mobiles (Smartphones et tablettes) pousse les entreprises à proposer toujours plus de services aux utilisateurs. Elles ont, en effet, bien compris tous les enjeux qui se trament derrière les applications mobiles : communication améliorée, nouveaux services répondant aux exigences des utilisateurs, avantage concurrentiel, etc.

Cet intérêt pour les technologies mobiles offre de nouvelles perspectives de marché, notamment pour les entreprises éditrices (que ce soient des Mobile Agencies, des Web Agencies) qui peuvent proposer leur expertise de développement. Depuis 2011, les nombres statistiques montrent que le marché des ventes de mobile surpasse les ventes de PC et donne donc des perspectives de marché de développement d'applications mobiles très importantes.

Les développeurs sont obligés de s'adapter aux plateformes de développement mobile qui se diffèrent de celle des applications conçues pour les PC. En effet, ces plateformes imposent des contraintes fortes, les ressources étant limitées : taille d'écran réduite, processeur faible, stockage limité, connexion lente et intermittente, RAM limitée. Toutes ces limitations impliquent une discipline de développement particulière pour que l'expérience utilisateur soit toujours efficace.

Nous avons discuté dans le chapitre 1 que de multiples systèmes d'exploitation mobiles (OS) (Android, iOS, BlackBerry, Windows Phone) se disputent le marché de la mobilité. En outre, à chaque OS correspond une plateforme de téléchargement d'applications (ex. Play store pour les applications Android) des outils de développement et un langage de programmation permettant le développement et la distribution des applications.

Cette hétérogénéité des outils de développement et des langages rend difficile le développement d'applications mobiles multiplateformes. Elle pousse les développeurs à faire un choix sur la plateforme, tout en assurant la plus grande distribution possible (Dupont 2013)

3.4.1 Des outils pour aider les développeurs ?

Google a récemment apporté un élément de réponse avec le projet J2ObjC qui permet de convertir du code Java (Android) en code Objective-C (iOS). L'intérêt d'une telle démarche est de pouvoir récupérer le code fonctionnel d'une application Android afin de faciliter la conversion vers la plateforme iOS. Mais cette approche pose encore de nombreuses questions:

- Qu'en est-il de la conversion Objective-C vers Java ? Et vers d'autres OS ?
- Et plus généralement, comment capitaliser sur le code métier d'une application indépendamment des préoccupations techniques afin d'en faciliter la migration ?

3.4.2 Une réponse avec le MDA

L'approche MDA a su faire ses preuves pour le développement d'applications d'entreprise et peut également apporter beaucoup pour les applications mobiles. Au travers de 5 points, nous allons voir pourquoi et comment l'approche MDA peut nous aider à *assurer la pérennité des savoir-faire*, le *gain de productivité* tout en répondant aux problématiques de *fragmentation des plateformes mobiles*.

3.4.2.1 Réduire les problématiques de fragmentation

Chaque OS mobile détient une part du marché, et donc chacun possède sa communauté d'utilisateurs et de développeurs. Cette fragmentation pose un souci pour les entreprises désireuses de toucher un large public, car le développement des applications multiplateformes est très coûteux en temps et en argent. Les entreprises se trouvent face à un dilemme :

- Développer une application de qualité en un temps efficace et limiter sa distribution à une plateforme spécifique.
- Développer une application aux fonctionnalités réduites, mais proposées sur plusieurs plateformes et à un large panel d'utilisateurs.

Des solutions hybrides existent comme Cordova (application native construite à partir de vue Web et Html5) ou Flex 4.6 (exécution sous plateforme Flash) et permettent d'assurer du multiplateforme, mais leur prise en charge reste techniquement partielle. En réalité, une application native restera plus adaptée pour plusieurs raisons :

- L'utilisateur dispos d'un environnement adapté à sa plateforme et à ses spécificités: l'expérience utilisateur reste optimale.
- Le développeur tire profit de l'architecture spécifique et optimise l'application en fonction de la plateforme.
- L'application peut évoluer et éventuellement tirer parti des fonctionnalités spécifiques à une plateforme ou OS.

L'approche MDA peut répondre à ces problématiques : décrire les besoins fonctionnels d'une application indépendamment de la plateforme d'exécution, et respecter les spécificités techniques de chaque plateforme. MDA permet de générer le code sans valeur ajoutée (boilerplate code) spécifique à chaque plateforme. On assure ainsi un gain de productivité, même sur des applications natives (Dupont 2013).

3.4.2.2 Capitaliser sur les concepts proches

Chaque OS mobile utilise son propre langage de programmation et arbore ses propres APIs mais les concepts généraux restent standards et communs. On peut se permettre de capitaliser des concepts communs entre différentes plateformes permettant de modéliser une application de manière générique, citons par exemple :

- **Activité:** correspond à une page de l'application, elle est une notion commune à tous les OS. Une application est composée d'une ou de plusieurs activités.
- **Gestion des données de l'application:** en séparant le code de l'interface graphique et le code fonctionnel.
- **Vues et actions associées:** une vue est un objet graphique par lequel l'utilisateur interagit avec l'application ce qui provoque une action liée à la vue. Nous retrouvons la plupart des vues sur tous les OS mobiles : bouton, saisi de texte, label, barre de navigation, bouton radio, case à cocher, etc.
- **Type d'évènements:** une application doit prendre en charge certains évènements susceptibles d'apparaitre durant son exécution tels que les appels téléphoniques, les SMS, etc.
- **Applications natives:** les applications natives telles que le calendrier, les contacts, les alarmes, le lecteur média sont communes à tous les OS.

L'approche MDA divise la conception de l'application en deux points de vue :

Un point de vue « fonctionnel » indépendant des détails liés à la plateforme d'exécution. On définit ce que l'on veut générer (le quoi). Dans cette partie, on modélise des notions abstraites telles que des écrans, boutons, objets métier, DAO, mais en restant indépendant de leur implémentation technique. Modéliser le fonctionnel de l'application assure la pérennité des savoir-faire.

Un point de vue « technique » dans lequel sera décrite l'architecture de la plateforme d'exécution. On définit comment on veut le générer (le comment). Dans cette partie, on spécifie comment la notion fonctionnelle doit être générée : quel langage (Objective C, Java), quelle version (Android 4.1 par exemple), quelle implémentation de BDD. Modéliser l'architecture technique assure la prise en compte de la plateforme d'exécution.

L'étape qui suit la description fonctionnelle et technique d'une application est la génération du code source, c'est le rôle du générateur (ex. Mia-Studio). Il s'appuie sur le modèle fonctionnel et technique de l'application pour générer du code source que le développeur pourra enrichir manuellement (le traitement algorithmique et les

ressources qui ne peuvent pas être déduits du modèle). La modélisation nous assure une *pérennisation du savoir fonctionnel.* La génération de code assure un gain de temps considérable entraînant *un gain de productivité.*

3.4.2.3 Élargir l'écosystème mobile

L'écosystème est tout ce qui vit autour d'une plateforme pour en faciliter le développement existant sous JavaEE/.NET est large, mais sur plateforme mobile, cet écosystème est encore réduit. L'utilisation de librairies tierces dans les applications mobiles est limitée par les différents OS pour plusieurs raisons :

- *Une taille d'application importante:* l'utilisation massive de librairies est contraire aux bonnes pratiques pour le développement mobile (problématique d'espace mémoire).
- *Corollaire du point précédent:* une taille d'application trop importante peut être un frein à son téléchargement (débit limité et frais des réseaux 3G).
- Le manque de visibilité des développements des librairies. Alors que les librairies pour applications JEE/.Net sont portées par de grands éditeurs (JBoss (Red Hat), Spring (Spring Source), Microsoft, etc.), les librairies mobiles existantes sont quasi exclusivement développées par des développeurs tiers. Le suivi et la maintenance ne sont donc pas toujours assurés.
- Les éditeurs Google, Apple, Microsoft gardent la main sur les évolutions des SDK et ne garantissent donc pas la compatibilité des librairies tierces avec les nouvelles versions des plateformes.

Pour assurer la pérennité du code et élargir son périmètre de compatibilité à plusieurs versions, il est donc préférable d'être indépendant au maximum des librairies tierces et d'avoir un code standard au SDK. L'approche MDA réduit le temps de développement en générant du *code standard* au SDK, en respectant les *bonnes pratiques d'architecture et de qualité.*

3.4.2.4 Industrialiser le développement

La mise en place d'une approche MDA, conjointement avec les différents composants d'une usine logicielle (construction, Intégration continue, analyse qualimétrique), industrialise le développement et assure une bonne qualité au code. L'industrialisation avec le MDA nécessite plusieurs éléments clés :

- La mise en place d'un POC par une équipe d'architectes logiciels. Ce POC servira de base de code attendu pour faciliter la réalisation du générateur.
- Un code généré ne doit pas être source d'erreurs et doit rester compatible avec les niveaux de qu'altimétrie logicielle (qualité, architecture).
- Une formation des développeurs à l'utilisation du DSL et du générateur.

- Un support technique permettant d'accompagner les développeurs sur les outils afin d'assurer un bon démarrage du projet.
- Une communauté autour du générateur qui remonte les bugs. L'amélioration en continu du générateur profite aux développements en cours et aux prochains projets.
- Une capitalisation sur les bonnes pratiques de développement afin de développer de futures applications encore plus rapidement.

Sur le marché, il y a encore peu de développeurs avec des compétences « mobile ». L'industrialisation des développements mobile par l'approche MDA, a encore plus de sens dans ce contexte, car il permet d'avoir un meilleur accompagnement et un meilleur suivit.

3.4.2.5 Réduire le coût de possession

Avec des budgets toujours plus serrés, les entreprises veulent réduire le coût de possession de leurs applications. L'approche MDA, et plus spécifiquement le DSL et le générateur de code ont une grande valeur de capitalisation et de réutilisabilité. Elles permettent ainsi de réduire les coûts de possession, notamment sur l'investissement initial et le coût de maintenance :

- Une conception facilitée par un DSL adapté, cadrant les besoins spécifiques aux plateformes mobiles.
- Un générateur de code disponible « sur étagère » et utilisable rapidement permet de booster le démarrage d'un projet.
- Réduction du support technique : le code ou les patterns complexes et qui sont enclins aux erreurs (typiquement liés à la persistance, mapping ORM) sont générés.
- Un code de qualité suivant les bonnes pratiques assure une meilleure maîtrise et une réduction des coûts de maintenance.
- Un processus itératif : l'outillage MDA par amélioration continue va réduire les coûts des projets progressivement au fur et à mesure de son amélioration.

Un OS peut aujourd'hui être leader du marché et demain perdre certains marchés. Un exemple, avec les annonces de Samsung, premier fabricant mondial de smartphones, qui compte sortir de nouveaux smartphones en 2013 en abandonnant Android pour Tizen. Ou encore à l'image du géant chinois Huawei qui propose son propre OS mobile ou encore plus récemment Firefox avec FirefoxOS.

En conclusion, au lieu d'investir sur la technologie qui abrite une application, mieux vaut investir dans une technique de développement qui permet de capitaliser la fonctionnalité d'une application indépendamment de sa technologie. Une méthode qui intègre l'évolutivité et la pérennité dans son processus de développement comme l'approche MDA (Dupont 2013).

3.5 Discussion

Nous avons discuté dans la section précédente les avantages apportés par le MDA dans le processus de développement des applications mobiles. Il est important de prendre en considération que les applications mobiles à base de Cloud sont ubiquitaire et plus complique que les applications mobiles traditionnelles, alors il est conseillé de dévier notre champ de recherche pour répondre à une question importante est de savoir comment profiter de la modélisation pour améliorer les performances des applications mobiles à base de Cloud et rendre leur développement un processus sophistiqué et méthodique

3.6 Conclusion

Nous avons présenté dans ce chapitre le MDA dans ses grandes lignes. Nous avons vu que MDA utilise fortement les modèles et nous avons indiqué les différents types de modèles employés par MDA, à savoir CIM, PIM et PSM. Aussi nous avons mis l'accent sur l'absence de la modélisation dans le développement des applications mobiles traditionnelles et à base de Cloud et nous avons discuté comment introduire le MDA dans le développement des applications (traditionnelles ou à base de Cloud) peux être avantageux.

Il est à noter que, la simulation peut être vue comme une forme de modélisation, car les entités de système auront des équivalents qui simulent leurs comportements dans un environnement virtuel. Dans le chapitre suivant, nous allons étudier les simulateurs de l'environnement Cloud existants et étudier la possibilité de simuler les applications mobiles dans un environnement Cloud.

Chapitre 4

Simulation informatique d'un environnement Cloud

4.1 Introduction

Certains services traditionnels et applications à base de Cloud, y compris les réseaux sociaux, hébergement web, la diffusion de contenu, et traitement en temps réel des données instrumentées. Chacun de ces types d'application se diffère en termes de composition, configuration et les exigences de déploiement. Quantifier la performance de politiques de provisionnement (ordonnancement et allocation) dans un véritable environnement Cloud (Amazon EC2 (EC2 2010), Microsoft Azure (Chappell 2008), Google App Engine (GAE 2010)) pour différents modèles d'application dans des conditions transitoires est extrêmement difficile parce que: (i) Les Clouds présentent des demandes, des modèles d'approvisionnement, la taille des systèmes et des ressources (matériel, logiciel, réseau) variable (Calheiros *et al.* 2011). (ii), les utilisateurs ont des exigences en matière de QoS concurrents et hétérogènes. (iii) Les applications ont différentes performances, la charge de travail, et les exigences qui évoluent et changent dynamiquement. Par conséquent, ce qui rend la reproduction des résultats qui peuvent être invoqués, une opération extrêmement difficile. En outre, il est long et fastidieux de reconfigurer l'analyse comparative des paramètres à travers une infrastructure Cloud de calcul massive-échelle sur plusieurs séries de tests.

De telles limitations sont causées par les conditions qui prévalent dans les environnements Cloud qui ne sont pas dans le contrôle des développeurs de services applicatifs. Ainsi, il est impossible d'effectuer des expériences comparatives dans des environnements reproductibles, fiables et évolutifs avec des environnements de Cloud monde réel. Une alternative plus viable est l'utilisation d'outils de simulation. Ces outils ouvrent la possibilité d'évaluer l'hypothèse (demande d'étude comparative) dans un environnement contrôlé où l'on peut facilement reproduire les résultats. Les approches fondées sur la simulation offrent des avantages significatifs aux entreprises de IT (ou toute personne qui veut offrir ses services d'application à travers les Clouds) en leur permettant de: (i) tester leurs services dans un environnement reproductible et contrôlable; (ii) régler le système des goulots d'étranglement avant de déployer sur de vrais Clouds; et (iii) d'expérimenter avec différents mix de charge et scénarios de performance des ressources sur les infrastructures simulées pour développer et tester des techniques le provisionnement des applications adaptatives (Quiroz *et al.* 2009).

Nous présentons dans ce chapitre l'importance de la simulation de système à base de Cloud et fournir un état d'art sur les outils de simulation les plus connus dans ce domaine.

4.2 Définition de la simulation informatique

La simulation informatique, ou simulation numérique est une série de calculs effectués sur un ordinateur et reproduisant un phénomène physique. Elle aboutit à la description du résultat de ce phénomène, comme s'il s'était réellement déroulé. Cette représentation peut être une série de données, une image ou même un film vidéo.

Un simulateur peut réagir à des modifications de paramètres et modifier ses résultats en conséquence. Une simulation numérique peut représenter des phénomènes physiques complexes dont la description repose sur un modèle mathématique comportant des équations aux dérivées partielles. L'ordinateur résout alors ces équations numériquement en utilisant la méthode des éléments finis (Sim 2016).

4.3 Etat d'art sur les outils de simulation

Très récemment, il devient de plus en plus important d'étudier et d'analyser le comportement des applications à base de Cloud, comment elles sont déployées, et si les services Cloud et les données sont sécurisées. Comme l'expérimentation dans un environnement réel est coûteuse et consommatrice en temps, et non répétitive (Calheiros *et al.* 2013), il est souvent difficile d'analyser les problèmes de performances et de sécurité sur les environnements de Cloud réelles. Par conséquent, la modélisation et la simulation de la technologie deviennent de plus en plus populaires de côté industriel et académique. Il est bien connu que la technologie de modélisation et de simulation est une méthode qui est souvent utilisée pour analyser des problèmes complexes dans le monde physique (Wang *et al.* 2011).

Dans la littérature, des simulateurs de systèmes distribués traditionnels (Buyya and Murshed 2002), (Dumitrescu and Foster 2005), (Legrand *et al.* 2003) ne peuvent pas fournir l'environnement qui peut être directement utilisé par la communauté de Cloud Computing. Par conséquent, beaucoup d'œuvres (ex. (Belalem 2012), (Kliazovich et al. 2013), (Lim et al. 2009) et (Nunez *et al.* 2011) etc..) accordent beaucoup d'attention sur cette question, ainsi que des simulateurs comme (Kliazovich *et al.* 2012), (Sriram 2009), (Wickremasinghe 2009) conçu spécialement pour le Cloud place au cours des dernières années. Brièvement, ces simulateurs peuvent généralement être divisés en deux types: simulateurs juste fondés sur un logiciel, et simulateur basé sur le logiciel et le matériel.

Dans cette section, nous résumons les résultats existants sur la modélisation et la simulation de Cloud Computing que nous pouvons savoir au meilleur de notre connaissance. En outre, nous présentons une analyse basant sur trois aspects : les plateformes sur lesquelles le simulateur est fondé, le langage de programmation utilisé, et enfin si le simulateur se concentre sur le côté logiciel ou hardware.

4.3.1 CloudSim

CloudSim (Calheiros *et al.* 2011) est un nouveau Framework de simulation généralisée et extensible qui se caractérise par la modélisation et la simulation des

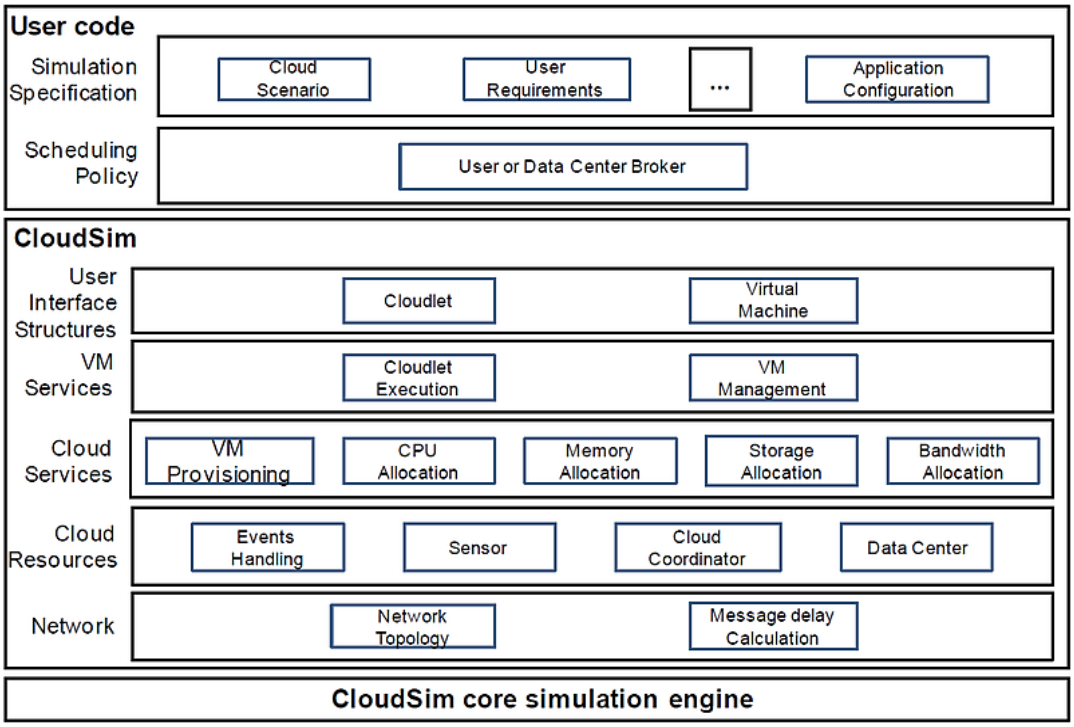
grandes infrastructures, y compris les centres de données qui hébergent plusieurs VMs. Ces dernières sont attribuées selon les différents politiques (en termes d'espace / et de temps), avec la possibilité de personnaliser les politiques en fonction des besoins de l'utilisateur.

4.3.1.1 Architecture de CloudSim

La figure 4.1 montre la conception multicouche de la structure logicielle de CloudSim et ses composants architecturaux. Les versions initiales de CloudSim utilisent SimJava comme le moteur de simulation à événements discrets (Howell and McNab 1998) qui prend en charge plusieurs fonctionnalités de base, telles que les files d'attente et le traitement des événements, création d'entités de système Cloud (services, Host, centre de données, Broker, MVs), la communication entre les composants et la gestion de l'horloge de simulation. Cependant, dans la version présentée, la couche SimJava a été supprimée afin de permettre des opérations avancées qui ne sont pas prises en charge par celui-ci.

La couche de simulation CloudSim fournit un support pour la modélisation et la simulation de centre à base de Cloud virtualisé y compris les interfaces de gestion dédiée aux machines virtuelles, la mémoire, le stockage et la bande passante. Les questions fondamentales, telles que le provisionnement d'hôtes pour les VMs, la gestion de l'exécution de l'application et la surveillance de l'état du système dynamique, sont traitées par cette couche. Un fournisseur de Cloud, qui veut étudier l'efficacité des différentes politiques d'allocation d'hôtes de machines virtuelles, aurait besoin de cette couche pour mettre en œuvre ses stratégies. Cette mise en œuvre peut être réalisée par l'extension programmée de la fonctionnalité de provisionnement de VMs. Il y a une distinction claire à cette couche liée au provisionnement des hôtes pour les VMs. Un hôte Cloud peut être en même temps alloué à un ensemble de machines virtuelles qui exécutent des applications basées sur les niveaux de QoS définis par les fournisseurs. Cette couche expose également les fonctionnalités qu'un développeur d'applications Cloud peut prolonger pour effectuer le profilage de la charge de travail complexe et étude de la performance des applications.

Le plus haut sommet de la couche dans la pile de CloudSim est le user Code qui expose les entités de base pour les hôtes (nombre de machines, leurs spécifications), les applications (nombre de tâches et de leurs exigences), les machines virtuelles, le nombre d'utilisateurs et de leur application les types et les politiques d'ordonnancement des courtiers. En étendant les entités de base au niveau de cette couche, un développeur d'applications Cloud peut effectuer les activités suivantes: (i) générer un mélange de demandes de la charge de travail des distributions, des configurations d'application; (ii) scénario de disponibilité de modèle de Cloud et effectué des tests robustes basés sur les configurations personnalisées; et (iii) mettre en œuvre des techniques de



provisionnement personnalisées pour les Clouds et leur fédération.

Figure 4. 1 L'architecture multicouche de CloudSim (Calheiros *et al.* 2011).

4.3.1.2 Implémentation du CloudSim

Dans cette section, nous fournissons les détails les plus fins en rapport avec les classes fondamentales de CloudSim, qui sont aussi les blocs de construction du simulateur. Le schéma global de conception de classe pour CloudSim est représenté sur la figure 4.2.

a) BWProvisioner

Ceci est une classe abstraite qui modélise la politique pour le provisionnement des bandes passantes pour les machines virtuelles. Le rôle principal de ce composant est de procéder à l'attribution des bandes passantes de réseau à un ensemble de machines virtuelles concurrentes qui sont déployées à travers le centre de données. Les développeurs et les chercheurs du système de Cloud peuvent étendre cette classe avec leurs propres politiques (priorité, QoS) pour tenir compte des besoins de leurs applications. *BWSimple Provisionning* permet à une VM de réserver autant de bande passante selon les besoins; cependant, ceci est limité par la bande passante totale disponible de l'hôte.

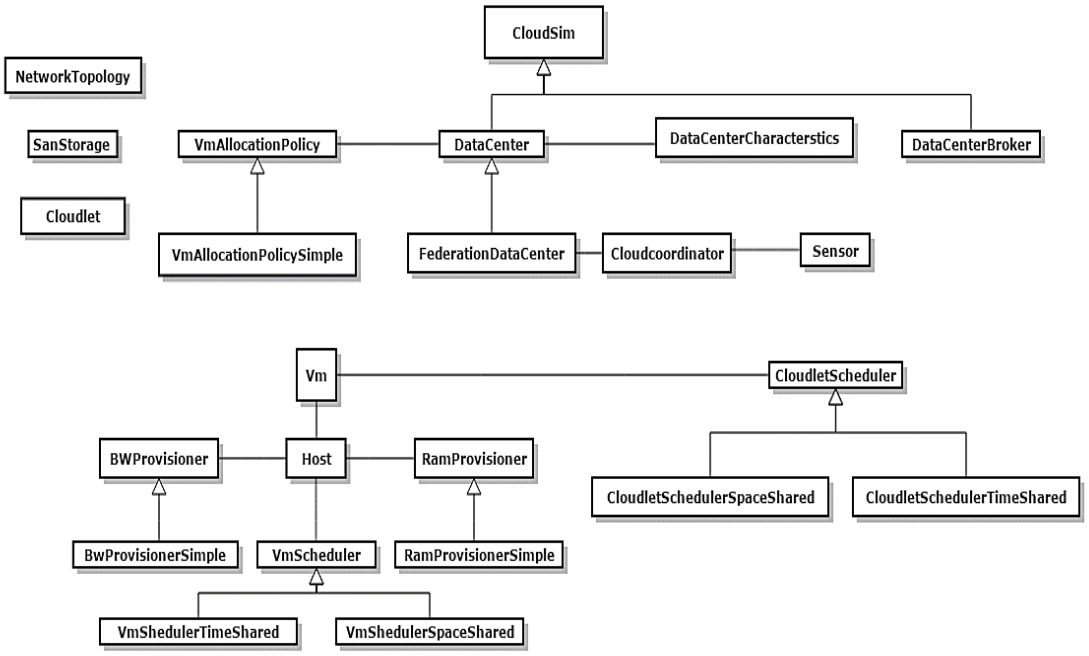


Figure 4. 2 Diagramme de classe représentant la conception de CloudSim.

b) CloudCoordinator

Cette classe abstraite étend un centre de données en Cloud à la fédération. Elle est chargée de surveiller périodiquement l'état interne des ressources de données du centre et sur cette base, il engage des décisions de déchiquetage de charge dynamique. La mise en œuvre concrète de cette class inclut les capteurs spécifiques et la politique qui devrait être suivie lors du déchiquetage de charge.

c) Cloudlet

Cette de classe modélise les services applicatifs basés cloud (ex. la livraison de contenu, les réseaux sociaux et flux de travail). CloudSim orchestrant la complexité d'une application en fonction de ses exigences de calcul. Chaque service d'application a un transfert de longueur d'instruction et les données préattribuées (à la fois avant et après récupère) les frais généraux qu'il doit entreprendre au cours de son cycle de vie. Cette classe peut également être étendu pour supporter la modélisation d'autres performances

et la composition des métriques pour des applications telles que les transactions dans les applications orientées base de données.

d) CloudletScheduler

Cette classe abstraite est étendue par la mise en œuvre des différentes politiques qui déterminent la part de la puissance de traitement entre cloudlets dans une machine virtuelle. Comme décrit précédemment, deux types de politiques de provisionnement sont proposés : partagé de temps (*CloudetSchedulerTimeShared*) et partagé espace (*CloudletScheduler SpaceShared*).

e) Datacenter

Cette classe modélise les services principaux au niveau des infrastructures (hardware) qui sont offertes par les fournisseurs de Cloud (ex. Amazon, Azure, App Engine). Elle encapsule un ensemble d'hôtes de calcul qui peut être soit homogène ou hétérogène par rapport à leurs configurations matérielles (mémoire, noyaux, capacité et stockage). En outre, chaque composant Datacenter instancie un composant de provisionnement d'application généralisée qui met en œuvre un ensemble de politiques d'allocation de bande passante, la mémoire et les périphériques de stockage pour les hôtes et les machines virtuelles.

f) DatacenterBroker or Cloud Broker

Cette classe modélise le Broker, qui est responsable de la médiation des négociations entre SaaS et les fournisseurs de Cloud; et ces négociations sont entraînées par exigences de QoS. Le Broker agit pour le compte des fournisseurs de SaaS. Il découvre les fournisseurs de services de Cloud appropriés en interrogeant la CIS et engage des négociations en ligne pour l'attribution de ressources/services qui peuvent répondre à la qualité de service de l'application a besoin. Les chercheurs et les développeurs de systèmes doivent étendre cette classe pour évaluer et tester les politiques de courtage personnalisé. La différence entre le Broker et le CloudCoordinator est que le premier représente le client (c.-à-d. décisions de ces composants sont réalisés en vue d'accroître les mesures de performance liées à l'utilisateur), alors que le dernier agit pour le compte du centre de données, à savoir qu'il essaie de maximiser la performance globale du centre de données, sans tenir compte des besoins des clients spécifiques.

g) DatacenterCharacteristics

Cette classe contient des informations de configuration des ressources des centres de données.

h) Hôte

Cette classe modélise les ressources physiques telles qu'un serveur de calcul ou de stockage. Elle encapsule des informations importantes telles que la quantité de mémoire et de stockage, une liste et le type de cœurs de traitement (pour représenter une machine multi-core), une répartition de la politique pour le partage de la puissance de traitement

entre les machines virtuelles, et les politiques de provisionnement de mémoire et bande passante pour les machines virtuelles.

i) NetworkTopology

Cette classe contient les informations pour induire le comportement du réseau (latences) dans la simulation. Elle stocke les informations de topologie, qui est générée en utilisant le générateur BRITE de topologie.

j) RamProvisioner

Ceci est une classe abstraite qui représente la politique de provisionnement pour allouer la mémoire principale (RAM) aux machines virtuelles. L'exécution et le déploiement de VM sur un hôte sont possibles uniquement si le composant *RamProvisioner* approuve que l'hôte ait la quantité requise de mémoire libre.

k) SanStorage:

Cette classe modélise un réseau de zone de stockage qui est communément ambiant dans les centres de données à base de Cloud pour stocker de grandes quantités de données (e.g Amazon S3, le stockage BLOB Azure). *SanStorage* implémente une interface simple qui peut être utilisée pour simuler le stockage et la récupération de toute quantité de données, sous réserve de la disponibilité de la bande passante. L'accès aux fichiers dans un réseau SAN au moment de l'exécution entraîne des retards supplémentaires pour l'exécution de l'unité de travail; cela est dû à des latences supplémentaires qui sont engagées dans le transfert des fichiers de données à travers le réseau interne du centre de données.

l) Sensor

Cette interface doit être mise en œuvre pour instancier un composant de capteur qui peut être utilisé par un *CloudCoordinator* pour la surveillance des paramètres de performance spécifiques (ex. Consommation d'énergie, l'utilisation des ressources). Rappelons que, *CloudCoordinator* utilise les informations de performance dynamique pour entreprendre des décisions d'équilibrage de charge. Les méthodes définies par cette interface sont: (i) définir les seuils maximaux et minimaux pour paramètre de performance et (ii) mettre à jour périodiquement la mesure. Cette classe peut être utilisée pour modéliser les services réels offerts par les principaux fournisseurs de Cloud tels que le CloudWatch d'Amazon et FabricController de Microsoft Azure. Un seul *DataCenter* peut instancier un ou plusieurs capteurs, chacun chargé de surveiller un paramètre spécifique de la performance du centre de données.

m) Machine virtuelle (*Vm*)

Cette classe modélise VM, qui est géré et hébergé par un composant *CloudHost*. Chaque composant *VM* a accès à un composant qui stocke les caractéristiques suivantes liées à une machine virtuelle: mémoire accessible, le processeur, la taille de stockage, et la politique de provisionnement interne des machines virtuelles qui est l'extension d'un composant abstrait appelé *Cloudlet Scheduler*.

n) VmmAllocationPolicy

Cette classe abstraite représente une politique de provisionnement qu'un moniteur VM utilise pour allouer les machines virtuelles aux hôtes. La fonctionnalité principale de la politique d'allocation Vm est de sélectionner l'hôte disponible dans un centre de données qui répond à la mémoire, le stockage, et l'exigence de disponibilité pour un déploiement VM.

o) VmScheduler

Ceci est une classe abstraite mise en œuvre par un composant hôte qui modélise les politiques (l'espace partagé, le temps partagé) requis pour l'attribution des noyaux de processeur pour les machines virtuelles. Les fonctionnalités de cette classe peuvent facilement être remplacées pour accueillir des politiques de partage de processeur spécifiques à l'application.

4.3.1.3 Quelques extensions de CloudSim

Plusieurs travaux ont été proposés pour étendre et améliorer CloudSim, en mettant l'accent sur la réduction et l'élimination des défaillances qui se produit lors de la soumission de l'emploi. Une autre extension de CloudSim est CloudAnalyst (Wickremasinghe 2009), (Wickremasinghe *et al.* 2010); le toolkit propose par Wickremasinghe est conçu pour étudier le comportement des applications d'Internet à grande échelle dans un environnement Cloud, aussi en offrant la possibilité d'exécuter plusieurs expériences de simulations en changeant tout simplement les paramètres. Garg et *al.* proposent dans (Garg and Buyya 2011) NetworkCloudSim cette extension de CloudSim prend en charge la modélisation en temps réel des DCs et généralise des applications comme l'e-commerce.

a) CloudAnalyst

Même si les Clouds rendent le déploiement des applications à grande échelle plus facile et moins cher, il crée aussi de nouveaux problèmes pour les développeurs. Parce que les infrastructures de Cloud sont distribuées, les applications peuvent être déployées dans différents lieux géographiques, et la répartition choisie peut avoir des impacts sur la performance de l'application pour les utilisateurs qui sont loin du centre de données.

Parce que les applications Internet sont accessibles par les utilisateurs à travers le monde, et parce que la popularité des applications varie le long du monde, l'expérience dans l'utilisation de l'application sera également variée. Quantifier l'impact du nombre d'utilisateurs simultanés, l'emplacement géographique des éléments pertinents, et le réseau dans les applications est difficile à réaliser dans des systèmes réels, en raison de la présence d'éléments qui ne peuvent pas être prédits ni contrôlés par les développeurs. Par conséquent, d'autres méthodes qui permettent la quantification de ces paramètres doivent être utilisées.

Pour permettre le contrôle et la répétabilité des expériences, des simulateurs tels que CloudSim sont utilisés. Des expériences de simulation appliquent des modèles des applications et des infrastructures (Gustedt *et al.* 2009). Ainsi, la simulation nécessite un

certain effort de développeurs d'applications pour modéliser à la fois l'infrastructure cible et le logiciel dans une langue qui est interprété par le simulateur. Même si les simulateurs offrent un soutien pour modéliser de tels scénarios, ils sont conçus pour être appliqués dans des expériences générales, et ainsi la modélisation de scénarios spécifiques peut être exigeante en temps (Wickremasinghe 2009).

L'un des principaux objectifs de CloudAnalyst est de séparer l'exercice de simulation d'expérimentation de l'exercice de programmation, donc un concepteur peut se concentrer sur la complexité de la simulation sans dépenser trop de temps sur les aspects techniques de la programmation. CloudAnalyst permet également un concepteur d'exécuter de façon répétée des simulations et de mener une série d'expériences de simulation avec de légères variations des paramètres d'une manière rapide et facile. Les principales caractéristiques de CloudAnalyst sont les suivantes.

Interface graphique : CloudAnalyst est équipé d'une interface utilisateur graphique facile à utiliser (voir la figure 4.3) qui permet aux utilisateurs de mettre en place des expériences rapidement et facilement.

Figure 4. 3 Interface graphique de CloudAnalyst .

La Possibilité de définir une simulation avec un degré élevé de flexibilité et configurable : la simulation de systèmes complexes tels que les applications Internet dépend de nombreux paramètres. En règle générale, les valeurs de ces paramètres doivent être arbitrairement prises ou déterminées par un processus d'essais et d'erreurs. CloudAnalyst fournit aux concepteurs un degré élevé de contrôle sur l'expérience, par la modélisation des entités et des options de configuration telle que: Data Center, dont la configuration matérielle est définie en termes de machines physiques composées de processeurs, périphériques de stockage, la mémoire et la bande passante interne.

Répétabilité des expériences : CloudAnalyst permet aux concepteurs de sauvegarder les paramètres d'entrée de la simulation expérimentée et des résultats sous la forme de fichiers XML de sorte que les expériences peuvent être répétées.

Sortie graphique : CloudAnalyst est capable de générer une sortie graphique des résultats de simulation sous forme de tableaux et de graphiques, ce qui est souhaitable pour résumer efficacement la grande quantité de statistiques qui sont collectées au cours

de la simulation. Une telle présentation efficace permet d'identifier les tendances importantes des paramètres de sortie et aide dans les comparaisons entre les paramètres connexes

Utilisation de la technologie consolidée et la facilité de l'extension : CloudAnalyst est basé sur une conception modulaire qui peut être facilement étendue. Il est développé en utilisant les technologies suivantes: Java (le simulateur est développé à 100% sur la plate-forme Java, en utilisant Java SE 1.6); Swing Java (le composant GUI est construit en utilisant des composants Swing); CloudSim (CloudSim dispose pour les centres de données de modélisation est utilisée dans CloudAnalyst); et SimJava (Howell and McNab 1998) certaines fonctionnalités de cet outil sont utilisées directement dans CloudAnalyst.

b) NetworkCloudSim

NetworkCloudSim (Garg and Buyya 2011)est un Framework de simulation qui prend en charge la modélisation des centres de données Cloud réels et applications généralisées telles que HPC, e-commerce et workflows. Le principal défi adressé est le développement de modèles d'application et de réseau qui sont suffisamment sophistiqués pour capturer les caractéristiques pertinentes du DCs Cloud réel, mais assez simple pour être prête pour une analyse. En particulier, les auteurs discutent des problèmes et des solutions ce qui concerne la modélisation de réseau interne d'un centre de données et des applications. La contribution ajoutée à CloudSim est la conception d'un modèle de flux de réseau pour les centres de données Cloud utilisant le partage de la bande passante et latences pour permettre des simulations évolutives et rapides. La plupart des paramètres du simulateur sont configurables, ce qui permet aux chercheurs de simuler une grande variété de topologies de réseau

(1) La conception et l'architecture de NetworkCloudSim

En dépit de plusieurs caractéristiques utiles de CloudSim, il ne peut être utilisé pour modéliser les centres de données complexes avec des modèles d'application différents, et les ressources hétérogènes tels que le traitement, stockage et le réseau. Pour rendre cet environnement plus réaliste, les caractéristiques et les problèmes liés à eux sont résolus.

Modèle d'application: Comme indiqué précédemment, la simulation des applications parallèles et distribuées n'est pas pleinement prise en compte dans la plupart des simulateurs informatiques actuels de Cloud. Bien qu'il soit possible de définir des emplois qui exigent plus d'un processeur, leur temps de simulation est simplement obtenu en leur mise à l'échelle d'un seul processeur. Cette approche ne convient pas dans des modèles plus sophistiqués d'application telle que MPI et de workflow. Les modèles d'application dans le Cloud Computing peuvent varier d'applications Web multi-niveaux tels que le commerce électronique aux applications scientifiques. Typiquement ces applications se composent de plusieurs tâches, qui communiquent les unes avec les autres. Une tâche se compose de quelques phases de traitement et de communication. Cela est vrai aussi bien pour les applications scientifiques et des applications web. Une application Web se compose de plusieurs niveaux, chaque niveau en cours d'exécution sur un serveur et la communication est entre ces différents niveaux.

La modélisation du réseau: La modélisation de topologies de réseau réel au sein du centre de données est une considération importante parce que la latence des données en raison de sur souscription de ressources telles que le réseau peut affecter la qualité du service offert au client Cloud. En outre, pour évaluer avec précision les applications avec des taches de communication et la migration de la machine virtuelle, il est également nécessaire de prendre en compte la topologie du réseau et la bande passante. En fait, dans de nombreuses applications, ceci est le facteur principal pour caractériser la performance.

(2) Implémentation de NetworkCloudSim

NetworkCloudSim étend les fonctionnalités de CloudSim f avec l'introduction de concepts qui modélise le comportement de l'application et le réseau interne d'un centre de données. Il y a trois acteurs principaux (ou entités) dans le NetworkCloudSim: *Switch*, *NetworkDatacenter*, et *NetworkDatacenterBroker*. La conception de NetworkCloudSim comme le montre la figure 4.4 comporte les éléments principaux suivants.

Switch : représente une entité de réseau qui peut être configuré comme un routeur ou un commutateur. Il peut modéliser des retards dans la transmission des données sur des hôtes ou un autre Switch en fonction de l'endroit où les données appartiennent. Actuellement, pour permettre la modélisation de diverses topologies, trois types de Switch sont modélisés : *root, aggregate* et *edge switch*. *Edge Switch* est connecté directement à des hôtes et à UpLinks connectés à un autre Switch. *Agregate Switch* a une UpLink et DownLink à des Switch. Le Root Switch est modélisé comme une entité de réseau qui est directement connecté à l'Internet/centre de données à « 'extérieur et est en DownLink vers d'autres commutateurs.

Network Packet et HostPacket: Ces classes représentent un flux de données d'une machine virtuelle à une autre dans un centre de données. Depuis sur chaque hôte les machines virtuelles sont connectées via un réseau virtuel créé par l'hyperviseur, le retard dans le transfert des données d'une machine virtuelle à un autre hébergé sur le même serveur est négligeable par rapport à lorsqu'elles sont transférées par l'intermédiaire véritable réseau du centre de données. Pour modéliser la différence entre ces deux réseaux, les auteurs ont conçu deux types de paquets: *HostPacket* et *Network Packet*. *HostPacket* est le paquet qui se déplace à travers le réseau virtuel. Considérant que *NetworkPacket* est le paquet qui se déplace d'un serveur à un autre. Chaque paquet

contient L'id de l'expéditeur et le récepteur des VMs, heure à laquelle il est envoyé et reçu et type de tâches.

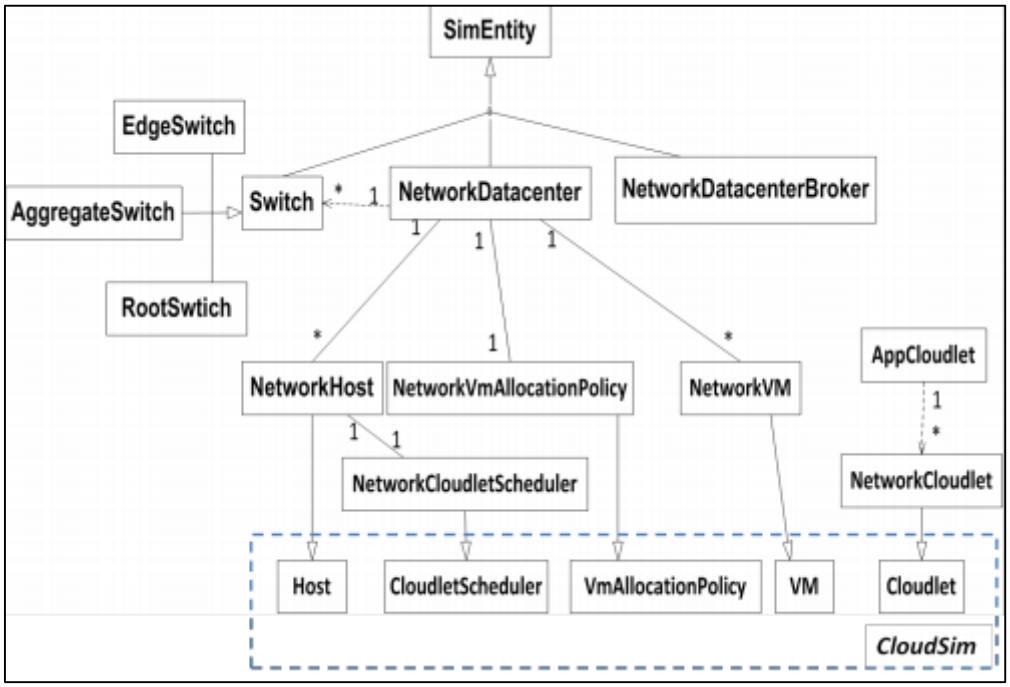


Figure 4. 4 Diagramme de classe présentant l'architecture de NetworkClousSim (Garg and Buyya 2011).

Pour rendre CloudSim conscient du réseau, toutes les classes clés sont également étendues. Par exemple, *NetworkHost* étend de la classe *NativeHost* et *NetworkVM* étend de la classe simple VM.

NetworkCloudlet: La classe Cloudlet, a été étendue pour représenter une tâche généralisée avec différentes étapes (TaskStage). Chaque étape peut être le traitement, l'envoi des données ou la réception des données. Cette classe contient également des informations de l'application à laquelle le Cloudlet appartient. Chaque *NetworkCloudlet* représente la plus petite entité d'exécution sur une machine virtuelle.

AppCloudlet: Représente une application avec des tâches multiples (NetworkCloudlet).

4.3.2 iCanCloud

Le simulateur d'infrastructure Cloud iCanCloud (Núñez *et al.* 2012) est conçu pour assurer les fonctionnalités comme la flexibilité, l'évolutivité, la performance et la convivialité. De plus, l'iCanCloud a été construit sur les principes de conception suivants : (1) il est ciblé de mener de grandes expériences, par opposition à des d'autres simulateurs; (2) il offre un hyperviseur mondial souple et personnalisable pour intégrer des différentes politiques de; (3) reproduis les types d'instances fournis par une infrastructure Cloud donnée; et enfin (4) contient une interface graphique conviviale pour la configuration et le lancement de la simulation, qui va d'une seule VM à grand système Cloud composé de milliers de machines. Les politiques d'allocation des ressources pour les centres de données (par exemple, en temps partagé vs partagé espace);

4.3.2.1 Plateforme d'iCanCloud

iCanCloud est une plate-forme de simulation visant à modéliser et simuler des systèmes de Cloud Computing, qui est destiné aux utilisateurs qui traite étroitement ces types de systèmes. L'objectif principal d'iCanCloud est de prédire les compromis entre le coût et la performance d'un ensemble donné d'applications exécutées dans un matériel spécifique, puis de fournir aux utilisateurs des informations utiles sur ces coûts. Cependant, iCanCloud peut être utilisé par un large éventail d'utilisateurs, des utilisateurs actifs de base et les développeurs de grandes applications distribuées (Nuñez *et al.* 2011).

Les simulations de scénarios de Cloud Computing sont modélisées à l'aide d'un ensemble de composants existants fournis par iCanCloud; ils représentent le comportement des composants réels qui appartiennent à de véritables architectures comme les disques, les réseaux, les mémoires, les systèmes de fichiers, etc. Ces composants sont organisés hiérarchiquement dans l'entrepôt d'iCanCloud, qui constitue le moteur de la simulation. Outre la conception d'environnements simulés en utilisant des composants fournis par iCanCloud, de nouveaux composants peuvent être ajoutés à son entrepôt. En outre, iCanCloud permet un remplacement facile des composants par d'autres composants particuliers. Ces composants interchangeables peuvent différer dans le niveau de détail (pour faire un compromis entre la performance par rapport à la précision), dans le comportement fonctionnel du composant, ou les deux.

4.3.2.2 Fonctionnalité d'iCanCloud

Les fonctionnalités les plus remarquables d'iCanCloud comprennent ce qui suit :

1) Les Deux Architectures de Cloud existantes et non existantes peuvent être simulées et modélisées.
2) Le module Cloud Hyperviseur offre une méthode simple pour intégrer et tester les deux nouvelles et existantes politiques de courtage.
3) Des Machines virtuelles personnalisables peuvent être utilisées pour simuler rapidement des systèmes uni-core / multi-core.
4) iCanCloud fournit une large gamme de configurations pour les systèmes de stockage, qui comprennent les modèles pour systèmes de stockage locaux, des systèmes de stockage à distance, comme NFS, et les systèmes de stockage parallèles, comme les systèmes de fichiers parallèles et les systèmes RAID.
5) iCanCloud fournit une interface graphique conviviale pour faciliter la production et la personnalisation des grands modèles distribués. Cette interface graphique est particulièrement utile pour: la gestion de l'entrepôt de VMs préconfiguré, de systèmes Cloud préconfigurés, l'expérience préconfigurés, lancer des expériences de l'interface graphique, et de générer des rapports graphiques. Cependant, les expériences peuvent être exécutées également en utilisant des scripts traditionnels de ligne de commande.

6) iCanCloud fournit une API POSIX et une bibliothèque MPI adaptée pour la modélisation et la simulation des applications. En outre, plusieurs méthodes pour la modélisation des applications peuvent être utilisées dans iCanCloud: utiliser les traces des applications réelles; en utilisant un graphe d'état; et la programmation de nouvelles applications directement dans la plateforme de simulation.

7) De nouveaux composants peuvent être ajoutés à l'entrepôt d'iCanCloud pour augmenter la fonctionnalité de la plateforme de simulation.

3.4.2.3 Conception d'iCanCloud

L'idée de base d'un système de Cloud Computing est de fournir aux utilisateurs un environnement matériel pseudo-personnalisable où ils peuvent exécuter un logiciel spécifique. Par conséquent, afin de modéliser des systèmes de Cloud Computing complet, l'architecture d'iCanCloud a été conçue sur la base de ce principe. Ainsi, la figure 4.1 représente l'architecture en couches d'iCanCloud (Castane *et al.* 2012).

La partie inférieure de l'architecture se compose de la couche de modèles de matériel. Cette couche contient essentiellement les modèles qui sont en charge de la modélisation des pièces de matériel d'un système, comme les lecteurs de disques, modules de mémoire et des processeurs CPU. En utilisant ces modèles, les systèmes distribués peuvent être modélisés et simulés. Dans ce qui suit, une présentation de : système de traitement (CPU), le système de mémoire, le système de stockage, et le système de réseau sont donnés.

Le module API du système de base est directement lié à la couche de modèles de matériel. Fondamentalement, ce module contient un ensemble d'appels système qui sont offerts comme une API (*Application Programming Interface*) pour toutes les applications exécutées dans une machine virtuelle modélisée en utilisant iCanCloud. Ainsi, les appels systèmes fournissent l'interface entre les applications et les services fournis par les modèles de matériel. En outre, les chercheurs peuvent écrire des applications à simuler dans iCanCloud utilisant cette API. Afin de maintenir un certain degré de compatibilité, cette API prétend être un sous-ensemble de POSIX.

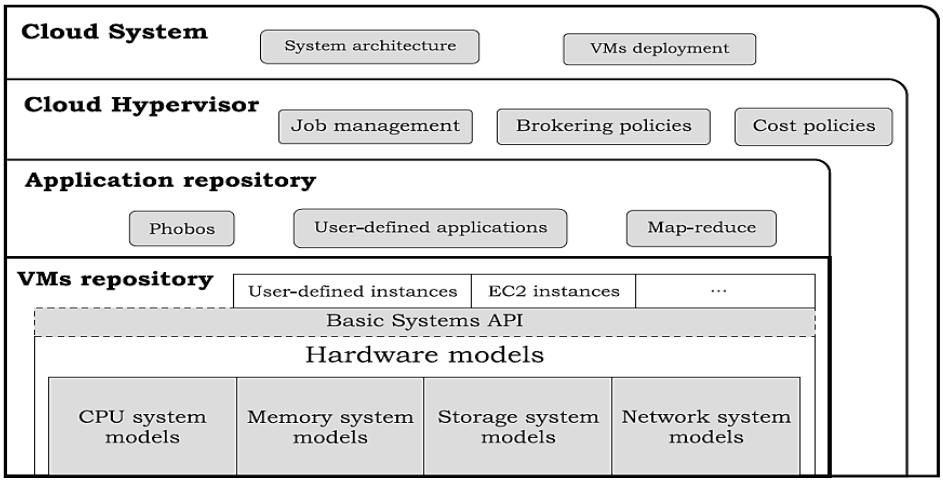


Figure 4. 5 L'architecture en couche d'iCanCloud (Nuñez *et al.* 2011)

La tâche principale du système de mémoire est d'attribuer le montant correspondant de la mémoire à chaque application qui l'exige. Ainsi, ce système reçoit des demandes d'allocation de mémoire et calcule où et comment cette mémoire doit être affectée . Cette fonction est très utile pour l'analyse de la quantité de mémoire utilisée pour chaque application, en particulier dans les environnements distribués.

Le système de stockage est chargé de gérer tous les accès aux données. Le système de réseau est en charge de la gestion des connexions avec d'autres applications situées dans des nœuds distants, et de traiter également les deux paquets reçus et envoyés.

La couche supérieure se compose d'un entrepôt de VMs. Ce dernier contient une collection de VMs préalablement définie par l'utilisateur. Dans un premier temps, le simulateur iCanCloud fournit quelques modèles de VMs existante dans Les Clouds bien connus comme Amazon (EC2). De plus, les utilisateurs peuvent ajouter, modifier ou supprimer des machines virtuelles à partir de cet entrepôt. Chaque VM est modélisé en configurant les modèles matériels fondamentaux correspondants pour chaque système de base.

La couche supérieure, appelée Cloud Hyperviseur, se compose d'un module en charge de la gestion de tous les travaux entrants et les instances de machines virtuelles où ces travaux sont exécutés. Une fois qu'un travail termine son exécution, ce module se met à désactiver les machines virtuelles où ce travail a été exécuté, et puis réaffecter les ressources disponibles dans le système pour exécuter les travaux restants. Ce module contient également des politiques de coûts afin d'assigner les tâches entrantes à une instance spécifique calculée par l'heuristique correspondante.

Enfin, au sommet de l'architecture est le module de Cloud System. Ce module contient une définition du système de Cloud complet, qui consiste essentiellement de la définition de Cloud hypervisor, et la définition de chaque VM qui compose le système.

4.3.3 Open Ciruss

Open Cirrus (Grossman *et al.* 2009) est un testbed de Cloud Computing qui se diffère des alternatives existantes, par la fédération des centres de données distribués. Il vise à stimuler l'innovation dans la recherche sur les systèmes et applications et catalyser le développement d'une pile de service open source pour le Cloud. Établi par Yahoo et HP, Open Cirrus (Campbell *et al.* 2009), (Avetisyan *et al.* 2010) fournit un testbed pour la simulation des centres de données distribués hétérogènes pour les systèmes, les applications et les services. Open Cirrus vise à encourager la recherche au niveau du système dans le Cloud Computing, encourager la recherche de nouvelle application à base de Cloud en proposant une plateforme pour les applications du monde réel, et de fournir un ensemble de données expérimentales afin de soutenir les chercheurs avec leurs expériences de haute qualité.

4.3.3.1 Objectifs d'Open Cirrus

Open Cirrus à quatre objectifs principaux.

Premièrement, le projet vise à encourager la recherche au niveau des systèmes dans le Cloud Computing. Dans l'environnement actuel, seuls les gros fournisseurs de services tels qu'Yahoo!, Google, Amazon et Microsoft ont accès à des centres de données distribués à grande échelle pour développer et tester de nouveaux systèmes et services. La plupart des chercheurs de Cloud Computing doivent généralement compter sur des simulations ou de petits groupes. Open Cirrus vise à contribuer à démocratiser l'innovation dans ce domaine en fournissant deux caractéristiques uniques essentielles à la recherche au niveau des systèmes.

- Les sites Open Cirrus permettent l'accès aux ressources matérielles et logicielles de bas niveau, par exemple, installer OS, accéder aux fonctionnalités matérielles d'accès.
- Le testbed comprend des sites hétérogènes dans différents domaines administratifs à travers le monde, ce qui permet aux chercheurs de profit de multiples centres de données.

Deuxièmement, Open Cirrus cherche à encourager de nouvelles applications à base de Cloud. Fournir une plateforme pour les applications et services dans le monde réel est une partie importante de l'Open Cirrus. Particulièrement intéressant sont le potentiel de développement de nouveaux modèles d'application et de les utiliser pour comprendre le soutien au niveau du système, et en utilisant la nature fédérée de l'Open Cirrus pour fournir une plateforme pour de nouveaux types d'applications et de services fédérés qui fonctionnent sur plusieurs centres de données.

Troisièmement, Open Cirrus offre une collection de données expérimentales. Les chercheurs manquent souvent des ensembles de données avec lesquelles effectuer des évaluations expérimentales de haute qualité. Les sites Open Cirrus permettront aux chercheurs d'importer, de stocker et de partager des ensembles de données à grande échelle tels que des robots Web et des traces de la charge de travail des centres de données. Avec de telles installations, Open Cirrus pourrait devenir un «trou d'eau» où les chercheurs ayant des intérêts similaires peuvent échanger des ensembles de données et de développer des points de repère standards de Cloud Computing.

Quatrièmement, Open Cirrus vise à développer des piles et des API open source pour le Cloud. Pour se répandre, le Cloud Computing nécessite une pile logicielle non propriétaire et des fournisseurs neutres. Open Cirrus servira de plateforme que la communauté open source peut utiliser pour concevoir, mettre en œuvre et évaluer ces codes et interfaces pour tous les niveaux de la pile de Cloud. Open source cherche à devenir la base d'une plus grande communauté ouverte de Cloud.

Les sites ouverts Cirrus travaillent ensemble pour fournir un seul testbed fédéré, par opposition à chaque bâtiment du site et l'exploitation de cluster, pour trois raisons :

- La collaboration sur un seul grand effort permettra d'atteindre un plus grand impact que les participants pourraient individuellement;

- Tests dans les différents environnements de site permettront d'améliorer la qualité des logiciels et des services; et
- La mutualisation des ressources permettra d'améliorer l'efficacité parce que les sites partagent les innovations.

4.3.3.2 Architecture d'Open Cirrus

Plusieurs choix architecturaux de haut niveau ont conduit à la conception de l'Open Cirrus.

La conception de l'architecture de services open Cirrus est guidée par le désir de créer une ressource unifiée et cohérente, plutôt que plusieurs clusters complètement disjoints.

a) Accès direct aux ressources physiques

La recherche sur les systèmes est prise en charge en permettant un accès direct aux ressources physiques. Par exemple, les chercheurs peuvent avoir des mots de passe, installez des images du noyau et des processeurs d'accès, chipsets, et le stockage. Cependant, certaines ressources, en particulier les ressources réseau nécessaires à la bonne isolation telles que les configurations virtuelles de réseau local (VLAN) de commutation, peuvent être virtualisées ou indisponibles.

b) Environnement d'exploitation similaire

Étant donné que plusieurs organisations ayant des pratiques différentes en gérant les sites Open Cirrus, il est possible que ces sites aient des environnements d'exploitation identiques. Cependant, il est possible de créer des environnements d'exploitation similaires en définissant un ensemble minimum de services que chaque site doit offrir.

c) Soutien de l'utilisation quasi-production.

Tout en soutenant la recherche du logiciel système de Cloud Computing est la mission centrale d'Open Cirrus, recherche robuste nécessite souvent l'accès à des cas d'utilisation du monde réel. Par conséquent, les sites Open Cirrus cherchent à offrir des services de haute qualité aux chercheurs qui ne sont pas nécessairement mènent des recherches au niveau des systèmes. Cette utilisation fournit les charges de travail, des traces, et les essais nécessaires pour un aperçu de l'utilisation du monde réel.

d) Services mondiaux disponibles à partir de tout site.

Un petit ensemble de services mondiaux sont disponibles à partir de tout site Open Cirrus. Des exemples comprennent le service mondial de l'authentification de connexion, la surveillance mondiale et un service de stockage modéré échelle pour les fichiers de configuration, des résultats intermédiaires ou binaires.

4.3.3.3 Architecture de pile de service

Typiquement les Sites Open Cirrus se composent de fondation, utilité et domaine de service primaire comme montré dans la figure 4.6

Zoni : le service de base de l'architecture logicielle est Zoni. Fondamentalement, Zoni est le composant logiciel responsable de la gestion des ressources physiques dans le cluster et est essentielle pour fournir aux utilisateurs l'accès au serveur pour effectuer des recherches de système logiciel. Ce composant fournit cinq fonctions principales:

- La répartition des nœuds serveurs;
- l'isolement des groupes de nœuds, appelés domaines;
- Le provisionnement des logiciels clés dans un domaine;
- La gestion des serveurs out-of-band
- Débuggant des nœuds alloués.

Services de domaine principal : Naturellement, pas tous les utilisateurs du cluster ne sont intéressés à la gestion d'un domaine Zoni; certains sont intéressés par le développement de services de niveau supérieur, et certains sont tout simplement intéressés par l'utilisation des services offerts. Pour servir ces deux derniers groupes d'utilisateurs, un domaine dans chaque site est désigné le domaine principal et fournit un ensemble stable de services pour une utilisation en production.

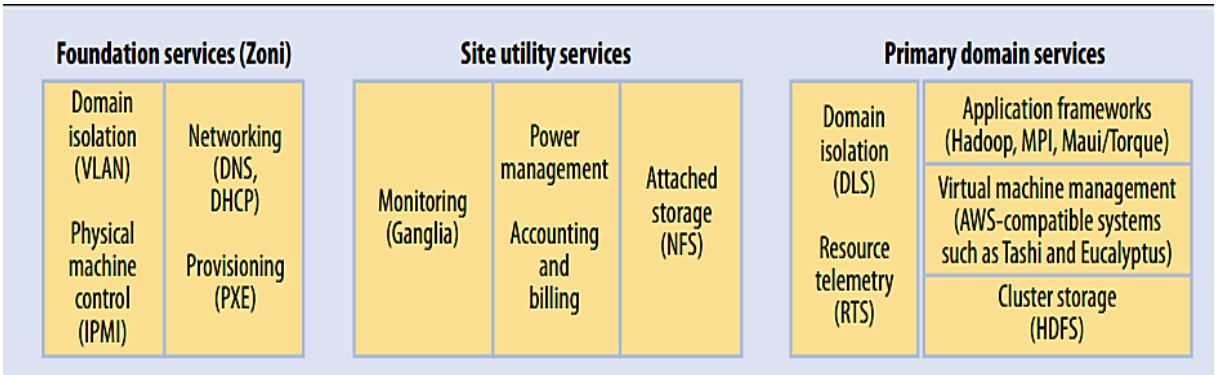


Figure 4. 6 Open Cirrus site services. A typical site consists of foundation, utility, and primary domain services (Avetisyan *et al.* 2010).

Services d'utilité du site : Pour gérer un site Open Cirrus facilement, de nombreux services supplémentaires, moins critiques sont nécessaires. Par exemple, un service de surveillance tel que Ganglia non seulement permet à l'administrateur du site pour surveiller le bon fonctionnement du Cluster, il facilite également la collecte des données opérationnelles de cluster qui peut informer les futurs projets de recherche. Certains systèmes de stockage de fichiers réseau classique sont pratiques pour stocker les scripts utilisateurs, petits ensembles de données, et de petits fichiers de sortie. Les services publics du site comprennent également des installations pour le suivi des ressources consommées par les utilisateurs et la gestion de la consommation d'énergie du Cluster

4.3.4 SPECI

Sriram propose dans (Sriram 2009) SPECI (*Simulation Program for Elastic Cloud Infrastructure*). SPECI est un toolkit de simulation basé sur SimKit (Buss 2002) qui permet l'exploration des aspects de la mise à l'échelle, ainsi que des propriétés de performance

des DCs, en offrant deux paquets principaux: l'un représentant la topologie des DCs, et l'autre englobe les composants pour l'exécution des expériences et de mesure.

4.3.4.1 Architecture de SPECI

La mise en œuvre de SPECI est divisée en deux paquets, on représente la configuration du centre de données et leurs topologies, et l'autre contient les composants pour l'exécution du test et de mesure.

La partie de l'expérience du simulateur se base sur SimKit, qui offre la programmation de l'événement, ainsi que le graphe de la distribution aléatoire. SimKit (Buss 2002) a des classes pseudo préconfigurées pour les distributions aléatoires, qui renvoient la même valeur pour les exécutions répétées. Cela permet d'exécuter de façon répétée avec des paramètres modifiés, et dans chaque exécution recevoir le même résultat graphique, et donc la même séquence d'ordonnancement et de planification pour les événements.

La classe d'entrée de simulation contient les configurations de tous les essais. Elle déclenche le moteur SimKit pour démarrer les simulations et gère l'analyse statistique de la production une fois que l'exécution a pris fin. Le moteur SimKit commence toujours simulations en appelant la méthode *doRun ()* de tous les objets existants dans le projet implémenté. Ceux-ci sont utilisés pour démarrer les expériences, et elles nécessitent de déclencher en conséquence un changement de paramètres configurés ou de placer de nouveaux événements sur le moteur de planification. Dans ce simulateur, il n'y a qu'une seule méthode *Run()* dans le *SingeltonHandler*. Cette méthode crée la configuration des DCs à partir du package DataCenterLayout avec les spécifications fournies. Elle ajoute ensuite trois types d'événements à EventScheduler: événements de sondage, les événements de mise à jour, et les événements d'échec. Le premier événement de sondage est généré à 2,0 secondes du temps de simulation pour permettre l'instanciation avant de mesurer d'éventuelles incohérences dans le système. Lorsque cet événement est déclenché, tous les abonnements sont testés pour des incohérences contre l'état réel et le nombre total transmis à une classe pour la collecte de statistiques de pointage du modèle de simulation. Avant de terminer l'événement de sondage, il se reporte à la prochaine exécution 1.0 seconde plus tard que le temps de simulation actuel. Ainsi, chaque seconde une sonde de contrôle est transmise pour une évaluation après la fin de la simulation. En outre, le gestionnaire génère un événement de mise à jour pour chaque nœud dans le centre de données. Cet événement déclenche le nœud pour qu'il mette à jour la liste de ses abonnements. Ces événements de récupération sont tirés d'une distribution uniforme avec un délai entre 0,8 et 1,2 seconde et se replanifient avec un retard de la même distribution. Le temps de l'échec suivant est un variable dans les expériences et doit être spécifié sous la forme d'une fonction aléatoire paramétrée. Lorsque l'événement d'échec est déclenché, il choisit un nœud au hasard dont il sera mis à avoir échoué. Si la fonction d'échec choisit un nœud qui est déjà échoué, il agira comme événement de réparation et réactivé le composant. Sinon, les composants défectueux ne sont pas réparés, et conservés jusqu'à ce que le conteneur entier de transport soit remplacé.

Le package *DataCenterLayout* contient des classes pour chaque type de composant dans le centre de données, telles que les nœuds et les liens du réseau. Ces composants imitent les opérations d'intérêt dans le centre de données observées, telles que le transfert de paquets de réseau, le maintien d'abonnements à d'autres nœuds, et en gardant les abonnements à jour en utilisant la politique choisie pour l'expérience. Les composants ont des points de contrôle qui peuvent être activés comme requis par l'expérience (Sriram 2009).

4.4 Classification

De nombreux outils de simulation de Cloud Computing ont été proposés dans la littérature. Dans cette section, comme indiqué dans le tableau 4.1, nous faisons l'analyse et la comparaison sur les simulateurs de Cloud Computing à partir de trois aspects: la plate-forme sous-jacente que les simulateurs sont basés sur, le langage de programmation utilisé par les simulateurs, le type d'entité de simulation offerte (hardware ou Software), et enfin, si le simulateur est ouvert pour de futures extensions ou non.

À partir du tableau 4.1, nous constatons que les simulateurs de Cloud Computing dans la littérature peuvent être classés en deux types, à savoir, des simulateurs conçus comme des logiciels et des simulateurs basés sur le logiciel et le matériel, et en attendant, la plupart de ces simulateurs sont des logiciels développés en Java. Il est à noter que CloudSim est le simulateur le plus étendu, vu qu'il est plus simple et facile à comprendre et manipuler.

Simulator	Underlying Platforms	Programming language	Software or Hardware	Support of Extension	Existing Extension
CloudSim	GridSim	Java	Software	Yes	Cloud analyst Network CloudSim
CloudAnalyst	CloudSim	Java	Software	yes	/
Network CloudSim	CloudSim	Java	Software	Yes	/
ICanCloud	SIMCAN	C++	Software	Yes	/
OpenCirrus	Federation of heterogonous DCs	/	Software and Hardware	Yes	/
SPECI	SimKit	Java	Software	Yes	/

Table 4. 1 Classification des simulateurs.

4.4.1 Discussion

Revenons au sujet principal de la thèse qui est la modélisation et la simulation des applications mobiles à base de Cloud, dans le chapitre précèdent nous avons mis l'accent

sur le fait que les applications mobiles traditionnelles et à base de Cloud ne sont pas considérées comme des applications assez compliquées pour avoir une représentation architecturale, dont nous ne sommes pas d'accord avec. Le même s'applique au domaine de la simulation. Nous attestons que, le comportement des applications mobiles dans un environnement Cloud doit être étudié de façon approfondie afin de développer des applications mobiles à base de Cloud performantes. Non seulement le développement de ces applications est couteux par ce qu'elles sont basées sur le Cloud qui offre des services a un prix donné, mais aussi le coût de développement et le coût des tests et les vérifications. Alors, dans le but de réduire le coût et augmenter le gain de marché des applications mobiles à base de Cloud, nous arguons que les deux phases de la modélisation et la simulation sont obligatoires avant le déploiement dans un environnement Cloud réel.

4.5 Conclusion

Le Cloud Computing a gagné une popularité assez importante dans les deux côtés (industriel et académique). Mais, il est impossible d'effectuer des expériences comparatives dans des environnements reproductibles, fiables et évolutives comme le Cloud. Une alternative plus viable est l'utilisation d'outils de simulation. Dans ce chapitre nous avons présenté les avantages apportés par la simulation en donnant un état d'art sur les simulateurs les plus connus dans la littérature. Une classification des simulateurs étudiée est établie et discutée. En discutant le domaine de la simulation dans la section 4.4, nous avons constaté que l'absence des simulateurs et les outils de modélisation dans le domaine de mobile Cloud Computing doivent être abordés, ce que fait le sujet d'une de nos contributions présentées dans le chapitre suivant.

Chapitre 5

Conception et Réalisation

5.1 Introduction

Le mobile Cloud Computing fut, apparaitre comme la solution la plus optimale qui répond, aux limites des appareils mobiles (à savoir : la durée de vie de la batterie, la consommation d'énergie et l'amélioration de la performance). Autant qu'un paradigme encore enfantin, il est évident, que d'autres limites restent à être traitées se manifestent. En outre, basé sur l'étude présentée dans les chapitres précédents qui porte sur le développement des applications mobiles traditionnelles et celles à base de Cloud, la modélisation et la simulation des applications mobiles à base de Cloud, nous avons identifié que :

- Les Applications mobiles à base de Cloud ne sont pas considérées comme autant des applications assez compliquées et sophistiquées pour avoir une représentation architecturale et être modélisées
- Plusieurs simulateurs pour étudier et analyser le comportement des applications dans un environnement Cloud sont proposés dans la littérature, mais malheureusement non un seul simulateur dédié aux applications mobiles à base de Cloud a été proposé;
- Les informations contextuelles sont très importantes pour améliorer la performance des applications mobiles à base de Cloud, mais elles ne sont pas pleinement exploitées.

En répondant aux questions posées précédemment, nous présentons dans ce chapitre, la conception et la réalisation d'un Framework conscient au contexte pour la modélisation et la simulation des applications mobiles à base de Cloud (*Context-Aware Framework For Modeling and Simulation of Mobile Cloud Applications*).

Deux principaux concepts dans notre Framework proposé doivent être implémentés à savoir : la modélisation et la simulation. L'élément de base dans notre projet de recherche c'est l'architecture de l'application mobile à base de Cloud (MC-App) dont la conception est faite via *Mobile Cloud Architecture Description Language* (MC-ADL) qui est implémenté sous forme d'un plug-in sous éclipse SDK en utilisant les technologies EMF et GMF. Nous avons affirmé précédemment que les informations contextuelles seront plus avantageuses si elles sont exploitées au niveau architectural mieux que le niveau middleware afin de bien raisonner sur eux. Pour cela, nous avons implémenté un plug-in toujours sous éclipse SDK pour les modéliser graphiquement. En travaillant dans l'environnement Éclipse, nous avons choisi CloudSim pour être la base de notre simulateur *Mobile Cloud Simulation toolkit* (MC-Sim) vu que la simulation et la modélisation des MC-Apps sont fonctionnellement liées. Le langage utilisé par CloudSim

est Java et par conséquent notre prototype y compris les algorithmes de configuration, d'évolution et de traitement des informations contextuelles sont programmées en utilisant Java afin de garder la cohérence entre les outils du Framework proposé.

Une partie de ce chapitre porte sur le concept architectural du Framework proposé. Nous allons entamer dans la section 5.2 les problèmes dans le développement des applications mobiles. Une représentation de l'architecture générale est donnée dans la section 5.3, ensuite nous détaillons les composants du Framework en termes d'architecture de chaque outil. Les détails de la mise en œuvre sont donnés dans les sections à venir et enfin un exemple pour évaluer le prototype du Framework est présenté comme étude de cas dans la section 5.10.

5.2 Développement des applications mobiles à base de Cloud

Avec la croissance des appareils mobiles, les entreprises semblent développer un intérêt particulier pour la mobilité afin de s'adapter au changement des utilisateurs. *Telerik*, une filiale de *Progress* ,a amorcé une enquête menée auprès de 3 000 professionnels en informatique pour faire un état des lieux de la mobilité.

L'enquête a révélé que 57% des développeurs étaient encore novices en matière de développement mobile ou encore n'avait jamais développé une application mobile. Parmi les 47% qui ont une expérience en développement mobile, ils livrent en moyenne une application fonctionnelle par an.

Parmi les facteurs identifiés comme des freins dans la livraison des applications dans les temps, 16% des développeurs ont avancé que le changement de technologie ou de pratiques de développement est des inhibiteurs, 19% ont estimé que le manque de temps était frustrant et 15% ont parlé de contraintes budgétaires.

La majorité des développeurs (43%) a expliqué que concevoir des applications mobiles pour une plus grande efficacité opérationnelle, 39% pour générer des revenus, 38% pour améliorer la productivité et 35% pour améliorer le service client.

(44%) des développeurs ont considéré l'expérience utilisateur comme étant l'élément le plus important à prendre en compte lors du développement d'applications mobiles de tout type. En seconde position figurait la facilité de la maintenance (24% du panel), la performance (15%) et enfin la sécurité (11%) (Stéphane 2015).

Comme n'importe quel produit ou service, l'application mobile a son propre cycle de vie. Le cycle de vie d'un logiciel (en anglais *software lifecycle*) désigne toutes les étapes à suivre lors du développement d'un logiciel de sa conception à sa disparition. L'objectif de cette séparation dans les étapes est de permettre de définir des repères intermédiaires permettant la validation de la conformité du logiciel avec les besoins définis et la vérification du procédé de développement. Le cycle de vie permet de détecter les erreurs au plus tôt et ainsi de maîtriser la qualité du logiciel, les délais de sa réalisation et les coûts associés. (CCM 2015).

Le cycle de vie d'une application mobile peut se représenter avec le schéma suivant.

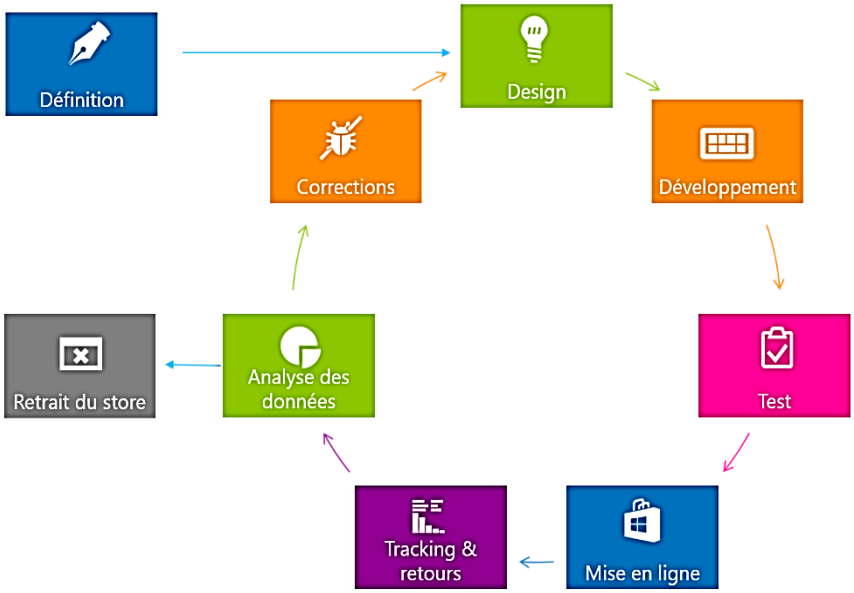


Figure 5. 1 Cycle de développement d'une application mobile

Nous remarquons dans la figure 5.1 que les applications mobiles partagent les grandes phases de développement avec les systèmes classiques, mais chacune de ces phases a ses propres particularités. Nous nous intéressons dans ce travail de recherche que par la phase de définition et de conception.

La phase de définition : Dans cette phase l'objectif principal de l'application, évidemment celui attendu par les utilisateurs, doit être fixé.

La phase de conception : Dans les systèmes classiques, cette phase consiste à décrire de manière non ambiguë, le plus souvent en utilisant un langage de modélisation, le fonctionnement futur du système afin d'en faciliter sa réalisation. Par contre, dans les applications mobiles c'est répondre aux questions concernant la technologie de développement (e.g application native, hybride ou web). Malheureusement, dans le développement des applications mobiles une phase de description de l'application d'une façon abstraite et générique n'existe pas.

D'après les résultats de l'enquête menée par Telerik et l'étude de méthode de développement ad hoc des applications mobile, nous constatons que :

Le développeur de l'application c'est lui-même le concepteur par contre aucun outil de modélisation ou de conception n'est utilisé. Avoir une vue abstraite sur l'application est indispensable pour cela il faut déléguer la conception de l'application à un concepteur et sa réalisation ou implémentation à un développeur expert pour éviter les freins mentionnés précédemment.

Il est à noter que, les applications mobiles à base de Cloud (MC-App) sont une nouvelle génération des applications mobiles et encore récentes pour être étudiées et analysées statiquement. Alors, les statiques sur le développement des applications mobiles traditionnelles sont applicables sur les MC-Apps.

Pour renforcer la phase de conception des MC-Apps, rendre leurs développements méthodiques et systématiques et produire des MC-Apps performantes, nous proposons un Framework pour les modéliser et les simuler. La conception générale du Framework proposé est donnée dans la section suivante.

5.3 Conception générale du Framework proposé

Notre vison en réalisant ce Framework est de rendre le développement des applications mobiles à base de Cloud un processus méthodique qui produit des applications performantes. Pour cela plusieurs propriétés doivent être assurées :

La généricité : Nous visons à définir un modèle générique favorisé par la méta-modélisation. En effet, la définition d'un méta-modèle permet de ne pas limiter notre approche à un simple langage ou une notation de modélisation. L'approche se place à un niveau d'abstraction supérieur aux langages et notations de modélisation. Nous nous intéressons principalement aux concepts utilisés pour décrire les architectures logicielles. Il s'agit de réifier ces concepts et de représenter explicitement leurs caractéristiques et les relations qui existent entre eux.

Une meilleure exploitation du contexte : Dans notre approche, nous considérons les informations venons de l'environnement mobile et de l'environnement Cloud comme deux types d'informations contextuelles très bénéfiques et essentiels pour garantir une qualité de performance très élevée. Pour cela, notre Framework doit fournir des informations contextuelles actualisées et à jour.

Mise à l'échelle : Le fait que l'application à développer est en relation fonctionnelle et directe avec un environnement élastique, elle doit être flexible pour l'ajout de nouveaux services et des nouveaux composants.

La pérennité : La durée de vie moyenne d'une application mobile ne dépasse pas trois mois. Par contre, une application ayant une bonne conception peut durer entre un an et un an et demi.

Capitalisation de savoir-faire : Chaque système d'exploitation mobile utilise son propre langage de programmation et arbore ses propres Apis, mais les concepts généraux restent standards et communs. Nous pouvons nous permettre de capitaliser des concepts communs entre différentes plateformes permettant de modéliser une application de manière générique (cf. section 3.4.2.2).

Afin de garantir les propriétés mentionnées précédemment, nous proposons l'architecture multicouche générale du Framework proposé (voir figure 5.2) comme suite :

5.3.1 Niveau architectural

Les applications mobiles à base de Cloud (MC-Apps) sont fonctionnellement liées avec le Cloud. Par conséquent, ces applications deviennent compliquées parce qu'elles doivent faire face aux obstacles posés par le Cloud comme l'hétérogénéité des services et

plateformes, l'intensivité des traitements, la variété des prix de service, la localisation géographique des centres de données et aussi les politiques de provisionnement. Pour réduire la complexité de ces applications et faciliter leurs mises en œuvre et leurs développements, nous attestons qu'une représentation architecturale est une étape obligatoire. Une vue architecturale sur les MC-Apps apporte les avantages suivants :

- Avoir une perspective complète et claire sur tous les composants (stables et variantes) entera dans le développement des MC-Apps.
- Favoriser le concept de la réutilisation et la capitalisation de savoir-faire. Capitaliser les concepts en commun entre différentes plateformes permettant de modéliser une MC-App de manière générique.
- Améliorer le processus de développement des MC-Apps multiplateforme et assurer le gain de productivité en écrivant une seule architecture indépendamment de la plateforme d'exécution et générer le code en ajoutant les spécifications techniques de chaque plateforme.
- Une caractéristique fondamentale dans le domaine du MCC est la mobilité qui met l'utilisateur et l'appareil mobile des sujets dans un environnement toujours changeant et offrant des informations contextuelles riches qui doivent être exploité intelligemment. Nous arguons qu'introduire ces informations dans l'architecture est plus bénéficiaire.

5.3.2 Niveau Middleware

La nature distribuée et hétérogène du Cloud pose plusieurs défis pour les développeurs des MC-Apps. Dans ce niveau, nous proposons un middleware Smart Cloud Gate (SCG) qui a les objectifs suivants en termes de défis à affronter :

MC-Apps Multi-Cloud : La popularité du Cloud a conduit les fournisseurs de services Cloud d'offrir des solutions sans cesse croissantes pour les consommateurs. Exploiter les particularités de chaque Cloud permettra d'améliorer la performance, la disponibilité et le coût des applications mobiles. Malheureusement, ces solutions sont généralement hétérogènes, incompatibles et posent certaines difficultés au développeur lors de l'élaboration et l'administration des applications mobiles multi-Cloud (c.-à-d. les applications mobiles qui utilisent les services fournis par les Cloud hybrides ou de différents services fournis par diffèrent Cloud). SCG offre des fonctionnalités de composition et d'adaptation pour prendre avantage du Cloud efficacement et satisfaire les exigences des applications qui nécessitent l'utilisation de plusieurs services à la fois.

Exploiter les informations contextuelles **:** retenus par l'aspect de la mobilité, les développeurs de MC-Apps doivent faire face à un environnement d'exécution qui est soumis à des changements constants. Ce dernier fournit des informations contextuelles qui permettront d'améliorer le développement de MC-Apps si elles sont utilisées de manière intelligente. La nécessité d'exploitation du contexte réside dans le fait qu'il fournit des informations importantes sur l'état récent des personnes, des lieux, des choses et des dispositifs voisinant (Dey 2001). Le principal défi est de tirer parti de

l'environnement changeant avec une nouvelle classe d'applications qui sont au courant du contexte dans lequel elles s'exécutent. Alors, SCG est un middleware conscient au contexte qui implémente les fonctionnalités d'acquisition, d'interprétation et exploitation des informations contextuelles.

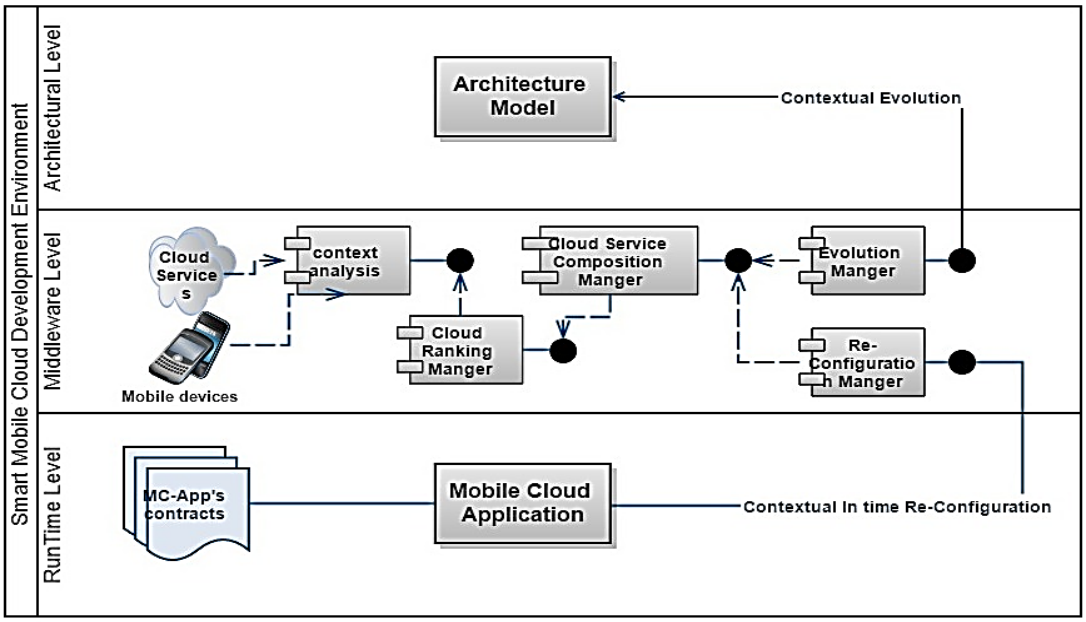


Figure 5. 2 Architecture générale du Context-Aware Framework for Modeling and Simulating Mobile Cloud Application.

Suivre les actualités récentes en Cloud : Le Cloud est une infrastructure élastique est évolutive qui accueille toujours une nouvelle vague de services, des utilisateurs et des fournisseurs. Pour garder l'application à jour et constamment prendre avantage du Cloud en termes de nouveaux services plus convenables en prix et en fonctionnalité offerts, SCG fournit des fonctionnalités de surveillance de Cloud, d'adaptation et de reconfiguration dynamique de l'application en temps d'exécution. Dans le but de maintenir la consistance entre le niveau architectural et le niveau application, SCG gère l'évolution architecturale simultanément avec la reconfiguration dynamique de MC-Apps en runtime.

2.5.3 Niveau Application

Avant de passer au développement des MC-Apps et leurs déploiements dans un environnement Cloud réel, une problématique très importante se manifeste qui est l'évaluation et le teste des MC-Apps dans un environnement à grande échelle. Afin de gagner du temps, étudier et analyser le comportement des MC-Apps dans un environnement gratuit et contrôlable, la phase de simulation est fondamentale avant le déploiement réel de l'application. Nous proposons dans ce niveau, un simulateur de comportement des MC-Apps dans un monde Cloud virtualisé et personnalisé selon les critères de développeur.

5.4 Processus de modélisation et de simulation

Le Framework proposé repose sur l'exécution collaborative des trois outils qui le compose (MC-ADL, SCG et MC-Sim). Le processus d'exécution comme illustré dans la figure 5.3 est initialisé au niveau de MC-ADL en décrivant successivement l'architecture de l'application et son contrat. Deux types d'objets sont générés de ces deux activités : l'architecture version graphique et version XML, le contrat version graphique et version.xml. Les versions XML sont générées dans le but de garder la cohérence entre les outils du Framework comme elles seront des entrées pour d'autres activités.

Dans l'architecture de MC-App, le concepteur définit d'une façon détaillée et objective tous les composants qui entrent dans l'exécution de l'application y compris ceux qui composent les deux environnements mobiles et Cloud. Dans notre Framework la phase de simulation a l'objectif d'étudier le comportement de l'application mobile dans un enivrement Cloud variant, pour cela il faut définir un comportement donné de l'application dans un état spécifique du Cloud simulé. Le comportement de MC-App est déjà prévu et défini dans le modèle contrat associé à son architecture alors que le comportement de l'environnement Cloud est exprimé sous forme de scénarios prédéfinis.

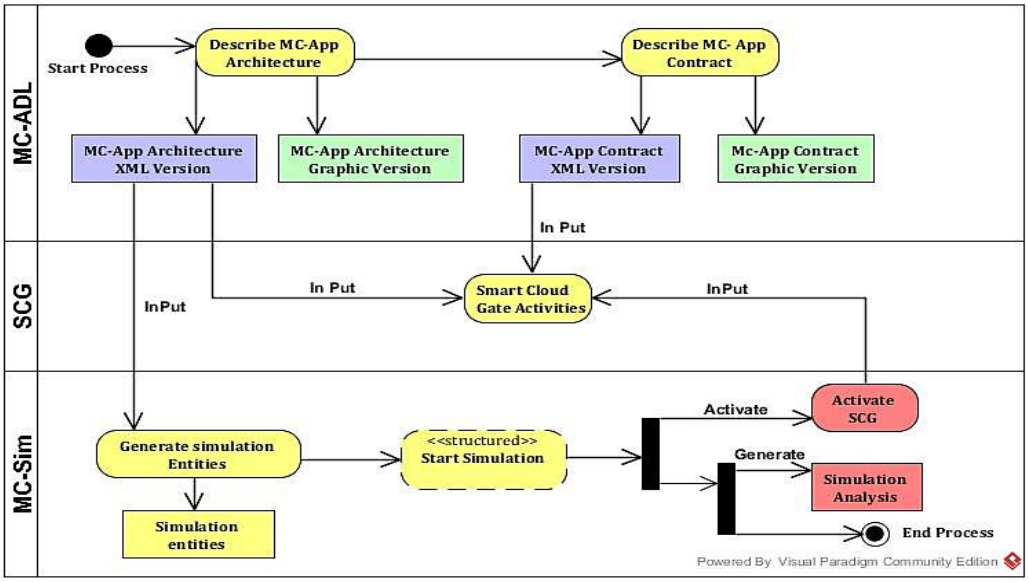


Figure 5. 3 Diagramme d'activité représentant le processus de modélisation et de simulation.

Afin d'étudier l'influence de chaque comportement sur l'autre, il faut d'abord éliminer l'hétérogénéité entre les deux modèles (architectural et de simulation). Alors, une génération des entités de simulation correspondant à l'architecture de l'application et son contexte mobile sont indispensables pour exécuter la simulation. Deux résultats peuvent être obtenus de la simulation :

- Si le MC-App garde son comportement prévu dans l'état de Cloud défini par le scénario, seules les analyses de la simulation seront générées.
- Sinon, le middleware Smart Cloud Gate sera activé pour exécuter les instructions définies dans le contrat afin de changer le comportement soit de

l'application soit changer l'état de l'environnement Cloud pour satisfaire les exigences de l'application.

5.5 Conception des applications mobiles à base de Cloud

L'objectif de l'architecture logicielle est d'offrir une vue d'ensemble et un fort niveau d'abstraction afin d'être en mesure d'appréhender les applications mobiles qui sont devenues de plus en plus complexes (Amirat 2010). Dans le cadre de cette thèse, nous nous intéressons spécialement aux architectures logicielles à base de composants. Ces dernières reposent sur une décomposition du système en un ensemble de composants qui communiquent entre eux par l'intermédiaire des connecteurs. Les concepteurs de logiciels ont besoin des langages de description d'architectures extensibles et des mécanismes flexibles permettant de manipuler les éléments et les concepts de base utilisés dans la description de l'architecture. Une architecture logicielle est construite à partir de composants, de connecteurs et de configurations avec des contraintes sur leurs assemblages ainsi que d'autres contraintes sur leurs comportements (Sadou-Harireche 2007).

5.5.1 Architecture des applications mobiles traditionnelles

Les appareils mobiles peuvent généralement fonctionner comme des Thin clients ou des Fat clients, ou ils peuvent être développés afin qu'ils puissent héberger des pages Web.

Les Thin clients n'ont pas de code d'applications personnalisées et comptent entièrement sur le serveur pour leurs fonctionnalités. Les Thin clients utilisent les applications web et les navigateurs de type Wireless Application Protocol (WAP) pour afficher les pages html5 ou XML.

Les Fat Clients ont généralement une à trois couches de code d'applications sur eux et peuvent fonctionner indépendamment d'un serveur pendant une certaine période de temps. Généralement, les clients Fat sont les plus utiles dans les situations où la communication entre un client et le serveur ne peut pas être garantie (voir figure 5.4) (J. Boccuzzi and M. Ruggiero 2011).

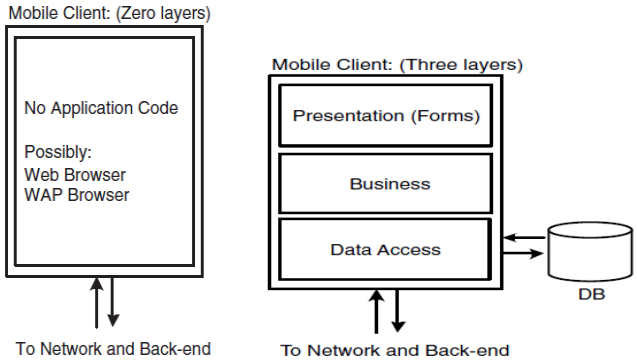


Figure 5.4 La structure des applications mobiles (J. Boccuzzi and M. Ruggiero 2011).

Par exemple, une application de type Fat Client peut être en mesure d'accepter l'entrée d'utilisateur et de stocker des données dans une base de données locale jusqu'à ce que la connectivité avec le serveur soit rétablie et les données peuvent être déplacées vers le serveur. Cela permet à l'utilisateur de continuer à travailler même s'il est hors de contact avec le serveur. Contrairement aux applications de type Thin Client, les Fat clients dépendent fortement du système d'exploitation et le type de dispositif mobile et le code peuvent être difficiles à libérer et à distribuer.

5.5.2 Architecture proposée pour les applications mobiles à base de Cloud

Les applications mobiles à base de Cloud se diffèrent des applications mobiles traditionnelles en déléguant le stockage et le traitement des données aux services Clouds. Par conséquent, leurs architectures aussi diffèrent. Nos proposant dans cette thèse l'architecture simplifiée d'une MC-App comme montré dans la figure 5.5.

MC-Apps sont similaire aux applications de type Fat Client en partageant une structure multicouche qui se compose d'une couche de donnée, une couche de code métier et une couche de présentation (c.-à-d. User Interface). Non seulement MC-Apps traitent un environnement hétérogène, mais aussi favorisent une structure hétérogène dont nous l'avons présenté sous forme de différents types de données et de composants de base. Les données des MC-Apps peuvent entre sauvegarder localement (ex. donné d'authentification) ou dans le Cloud (ex. application d'imagerie qui sauvegarde les images en Cloud).

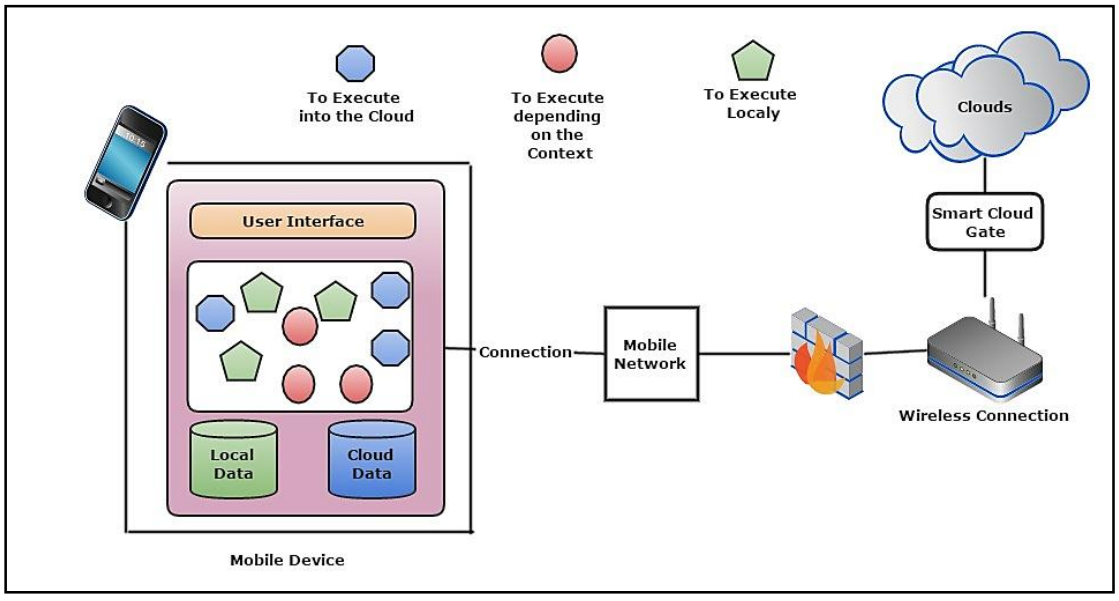


Figure 5. 5 Architecture de MC-App

Concernant le code de base de MC-App, il se compose de trois types de composants : les composants qui nécessitent une puissance de traitement significative sont exécutés en Cloud selon deux modèles d'accès notamment la délégation (Flores and Srirama 2013) ou le déchargement (Kumar and Lu 2010) (cf. section 2.8.2.1). Le deuxième type c'est les composants qui peuvent fonctionner normalement dans l'environnement local et enfin

nous avons ajouté un nouveau type de composants changeables que leurs exécutions dépendent fortement du contexte, soit du mobile soit du Cloud. Le changement est prévu dans *MC-App Contract* et exécuté par *Smart Cloud Gate*. Ce mécanisme sera expliqué dans les sections à venir.

En nous basant sur la définition des architectures logicielles données précédemment, nous arguons qu'on peut définir la description d'une simple application mobile. Mais malheureusement, cette définition ne convient pas à la structure distribuée, hétérogène et compliquée des applications mobiles à base de Cloud. Ces dernières nécessitent plus qu'un simple composant, connecteur et une configuration pour leurs descriptions, mais plutôt des nouveaux types de ces composants et distinguer précisément les relations entre chaque type. Nous présentons dans la section suivante le langage que nous proposons pour décrire les architectures des applications mobiles à base de Cloud.

5.6 Mobile Cloud Architecture Description Language (*MC-ADL*)

Un Architecture Description Language (ADL) est une notation standard pour représenter les architectures afin de contribuer à promouvoir une communication mutuelle, une concrétisation de la réalisation de décisions de conception précoce et la création d'une abstraction transférable d'un système. Les architectures dans le passé ont été largement représentées par le dessin de boîte et en ligne annotée avec des choses telles que la nature des composants, les propriétés, la sémantique des connexions et le comportement global du système. Les ADLs résultent d'une approche linguistique de la représentation formelle des architectures, et en tant que telle, ils abordent ses lacunes. En outre, les ADL sophistiqués permettent une analyse précoce et des tests de faisabilité des décisions de conception architecturale (ACCORD 2002).

Il est à noter que, les applications mobiles traditionnelles ou à base de Cloud sont développées d'une façon Ad-hoc sans suivre une méthode ou un processus de développement bien précis. On outre, nous avons mis en évidence précédemment le fait que l'architecture des MC-Apps se diffère des architectures logicielles traditionnelles. Par conséquent, les ADLs connus dans les littératures ne supportent pas la description des MC-Apps.

5.6.1 Conception du MC-ADL

Dans l'objectif de répondre aux manques résidants dans MCC à savoir : manque d'une présentation architecturale des MC-Apps et langage pour leurs descriptions, nous présentons notre ADL personnalisé MC-ADL. La figure 5.6 présente le méta-modèle simplifié de MC-ADL sous forme de diagramme de classes.

Afin de bien raisonner sur l'architecture des MC-Apps, il est nécessaire d'avoir une vue complète sur les environnements en sujet d'étude (le mobile et le Cloud). Conformément à l'architecture présentée dans la figure 5.5, l'application est fortement liée avec son environnement mobile et celui du Cloud. Nous favorisons le concept de la

généricité et l'indépendance de la plateforme d'exécution et les détails d'implémentation, mais cela n'empêche pas d'avoir une perspective initiale sur l'environnement d'exécution de l'application pour étudier son comportement. Pour cela, nous avons défini la classe abstraite ***mobile devise*** caractérisée par son contexte qui regroupe les propriétés ***intrinsèques*** de mobile et celle ***extrinsèque*** de son environnement. Il est à noter que le type de mobile y compris les technologies de mise en œuvre de l'application peut se varier tout au long de sa vie ; en d'autres termes, l'architecture proposée permet la définition des applications mobiles à base de Cloud multiplateforme.

L'application n'est pas connectée directement au Cloud, mais plutôt via un réseau mobile et réseau d'Internet. Les détails sur l'architecture de mobile Cloud Computing sont donnés dans la section 2.4. Nous avons ajouté une autre intermédiaire entre MC-Apps et le Cloud qui est un middleware conscient au contexte Smart Cloud Gate dont parmi ses fonctionnalités est de filtrer le Cloud afin de fourni une vue personnalisée selon les exigences de l'application.

5.6.1.1 Conception du côté Cloud

a) Définition du Cloud Computing

Le NIST (National Institute of standard and Technology) définit le Cloud comme suit : Le Cloud Computing est un modèle qui permet un accès pratique et à la demande réseau à un pool partagé des ressources informatiques configurables (ex. réseaux, serveurs, stockage, applications et services) qui peuvent être provisionnées rapidement et libéré avec un effort de gestion minimale sans interaction directe avec les fournisseurs des services. Ce modèle de Cloud est composé de cinq caractéristiques essentielles, trois modèles de services et quatre modèles de déploiement (Mell and Grance 2011).

b) Les caractéristiques d'un Cloud

Self-service à la demande*:* Un consommateur peut unilatéralement demander des capacités de calculs, telles que le temps de serveur et de stockage en réseau, automatiquement et sans la nécessité d'interagir avec les fournisseurs des services.

Accès au réseau: les ressources de calculs fournies sont disponibles sur le réseau et accessibles par le biais des mécanismes standards qui favorisent l'utilisation par les plateformes de type Fat ou Thin Clients (ex. les téléphones mobiles, tablettes ou ordinateurs).

La mutualisation des ressources: Les ressources informatiques fournies sont regroupées pour servir plusieurs consommateurs à l'aide d'un modèle multi-locataire avec des ressources physiques et virtuelles dynamiquement attribuées et rétribuées en fonction de la demande des consommateurs. Il y a un sens d'indépendance d'emplacement où le client n'a généralement aucune connaissance sur l'emplacement exact des ressources fournies, mais peut être en mesure d'indiquer l'emplacement à un niveau d'abstraction plus élevé (ex. pays, état ou centre de données). Exemples de ressources comprennent le stockage, le traitement, la mémoire et la bande passante réseau.

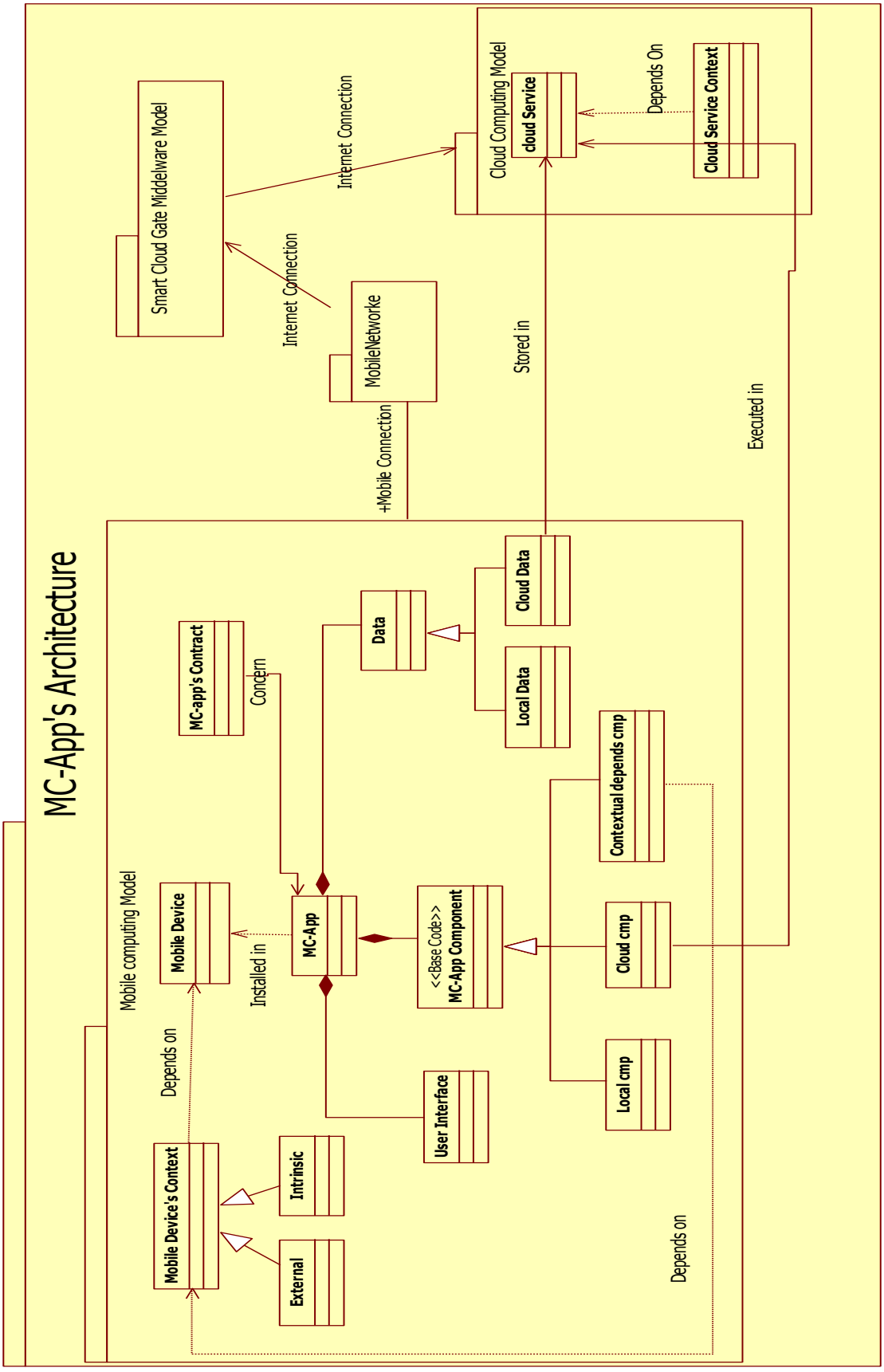


Figure 5. 6 Diagramme de classe présentant une vue simplifiée sur l'architecture de MC-App.

***Élasticité rapide**:* Des capacités peuvent être rapidement et de manière élastique provisionnée, dans certains cas, automatiquement et mise à l'échelle rapidement vers l'extérieur et vers l'intérieur en rapport avec la demande. Pour le consommateur, les capacités disponibles pour l'approvisionnement souvent semblent être illimitées et peuvent être appropriées en quantité à tout moment.

Service de mesure: Les systèmes de Cloud contrôlent automatiquement et optimisent l'utilisation des ressources en tirant parti d'une capacité de mesure à un certain niveau d'abstraction approprié pour des types de services (ex. le traitement, la bande passante et les comptes des utilisateurs actifs). L'utilisation des ressources peut être surveillée, contrôlée et rapportée en assurant la transparence tant pour le fournisseur et le consommateur du service utilisé.

Le Cloud n'est pas seulement une ressource de provisionnement de services à la demande via l'Internet, mais il a ces propres caractéristiques intrinsèques comme l'élasticité qui donne une illusion des ressources infinies et il favorise le parallélisme des tâches. Donc un Framework mobile Cloud parfait doit prendre avantage de toutes ces fonctionnalités. Malheureusement dans des travaux récents dans ce domaine (cf. Section 4), les approches proposées n'exploitent pas le Cloud d'une manière systématique et structurée. Aussi, le contexte du Cloud est géré séparément du contexte du mobile, pour cela les valeurs ajoutées du Cloud sur le mobile restent limitées. Le coté Cloud se compose d'un ensemble de Cloud de différents types (Private, Community, Public, Hybrid Cloud), des applications fournies aux clients sous forme de services à la demande (SaaS), des infrastructures comme services (IaaS) et des plateformes comme des services (PaaS). Plus détails sur le Cloud sont donnés dans la section 2.4.1. Chaque fournisseur de Cloud (IBM, Nist, Sisco, Microsoft, Amazone, etc.) a sa propre architecture référentielle de Cloud. Notre objectif et de prendre avantage de Cloud dans le niveau architectural de l'application mobile et non plus proposer une nouvelle architecture référentielle de Cloud. Les éléments que nous jugeons indispensable d'avoir des informations sur eux sont illustrés dans le modèle UML (Rumbaugh *et al.* 2004) (voir figure 5.7) qui représente le côté Cloud traité par notre Framework.

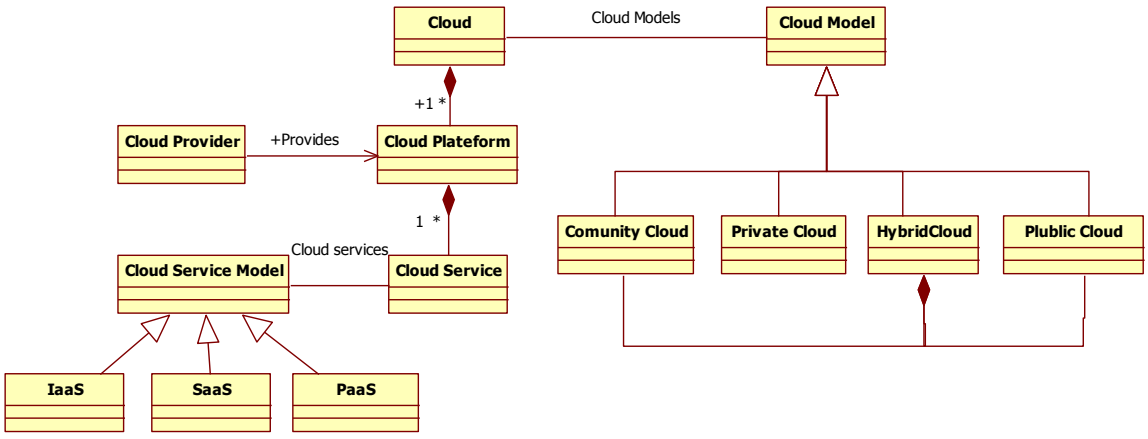


Figure 5.7 Diagramme UML représentons la modélisation du coté Cloud

Le modèle présenté en dessus doit répondre aux questions sur le Cloud, notamment quelques-unes: quel service le moins cher ? Quel centre de données le plus proche ? Quel service le plus performant ? Quel service avec le temps de réponse le plus optimisé ?

Ce raisonnement sur le Cloud est fait par le bai de *Cloud service Context*. Cette classe abstraite (Figure 5.6) englobe toutes les informations concernant le Cloud afin de répondre à n'importe quelle question sur lui.

5.6.2 Meta Modèle de Mobile Cloud Architecture Description Language

Dans la section précédente nous avons mis l'accent sur l'ensemble des entités que nous attestons indispensables pour décrire une application mobile à base de Cloud, l'organisation du ces entités et comment sont associées, et nous avons affirmé que les MC-Apps ont besoin de nouveau type de configuration de composant et de connecteur pour leurs descriptions. Le méta-modèle présenté dans la figure 5.8 introduit le nouveau type que nous proposons.

En utilisant le diagramme de classe, l'héritage et la composition de la notation UML (Rumbaugh *et al.* 2004), nous donnons une description des concepts du méta-modèle du MC-ADL. Le diagramme de classes focalise principalement sur le type de chaque concept et non plus sur sa description détaillée.

5.6.2.1 La configuration

La configuration peut être définie comme un graphe ayant l'objectif de décrire la façon dont les composants et les connecteurs sont reliés entre eux. L'assemblage de ces éléments dans une configuration est nécessaire pour savoir s'ils sont bien reliés, si leurs interfaces sont compatibles, si les connecteurs assurent une communication correcte et enfin étudier le comportement provoqué par la combinaison sémantique entre les composants. Faciliter la communication entre les différents intervenants dans le développement de MC-App est le rôle principal d'une configuration. Les configurations offrent une perspective abstraite du MC-App à un haut niveau qui peut être compressible par des personnes ayant de différents niveaux d'expertise et des connaissances techniques (ACCORD 2002).

La classe configuration est instançiable afin de favoriser la construction de différentes architectures des différentes applications. Ainsi, nous pouvons déployer une architecture donnée de plusieurs manières sans réécrire le programme de configuration/déploiement. MC-ADL définit trois types de configuration de plus celle des MC-Apps selon le type de composants qui l'englobent.

a) Configuration Cloud

Se compose des éléments de Cloud et les relations qui leurs relies. La réalisation de cette configuration peut être inspirée d'un Cloud réel (ex. Amazone Cloud Architecture) ou bien selon les préférences du concepteur ou le développeur.

b) Configuration Mobile

En revenons aux architectures de Mobile Computing, la configuration de côté mobile est partiellement stable et non changeable qui comporte la *configuration stable de Mobile network* et la *configuration changeable de l'application.* La configuration générale d'une application mobile se compose d'une *interface*, des composants qui représentent le *code métier* et enfin des composants de type *donné.*

5.6.2.2 Le composant

Un composant architectural décrit une unité de calcul ou de stockage de données à laquelle est attachée une spécification technique. Il peut représenter un petit programme simple, une grande application, un service ou un serveur. Le composant est l'entité principale dans une architecture, sa définition doit supporter la composition, l'indépendance descriptive, la réutilisabilité et il doit être autonome (Barbier *et al.* 2002).

Afin d'être réutilisable, le composant type permet d'encapsuler les fonctionnalités et les éléments internes. Le concepteur peut instancier un composant type plusieurs fois dans une même architecture ou le réutiliser dans d'autres architectures (Barais 2005). Évidemment, les instances d'un type partagent les mêmes structures et comportements que celui de leur type. La notion de typage facilite également la compréhension et l'analyse d'une architecture.

MC-ADL ajoute trois types de composants selon l'environnement sur lequel ces composants sont déployés en plus de ceux déjà définis dans les ADLs traditionnels (Composant simple et composant composite).

a) Composant-type Cloud :

Ce type de composant peut définir l'abstraction de plusieurs éléments de Cloud (ex. modèle de Cloud, type de service et type de plateforme).

b) Composant- type Mobile

Les éléments du Mobile y compris les composants de l'application et les réseaux mobiles entrent dans les sous types de ce composant.

c) Composant-type Contexte

Un élément très important entre dans l'architecture des MC-Apps c'est le contexte. MC-ADL définit deux sous types de ce composant.

Mobile Context *:* Ce composant définit toutes les entités qui entrent dans la composition du contexte mobile à savoir : les caractéristiques intrinsèques des appareils mobiles (ex. RAM, CPU, capacité de stockage, taille de l'écran) et les caractéristiques extrinsèques (ex. bande passante, connexion Internet et appareil voisinant).

Cloud Context *:* Un composant qui rassemble les préférences prédéfinies (ex. types de services, modèle de services, liste des prix désirés, etc.) du concepteur sur les services Cloud à utiliser.

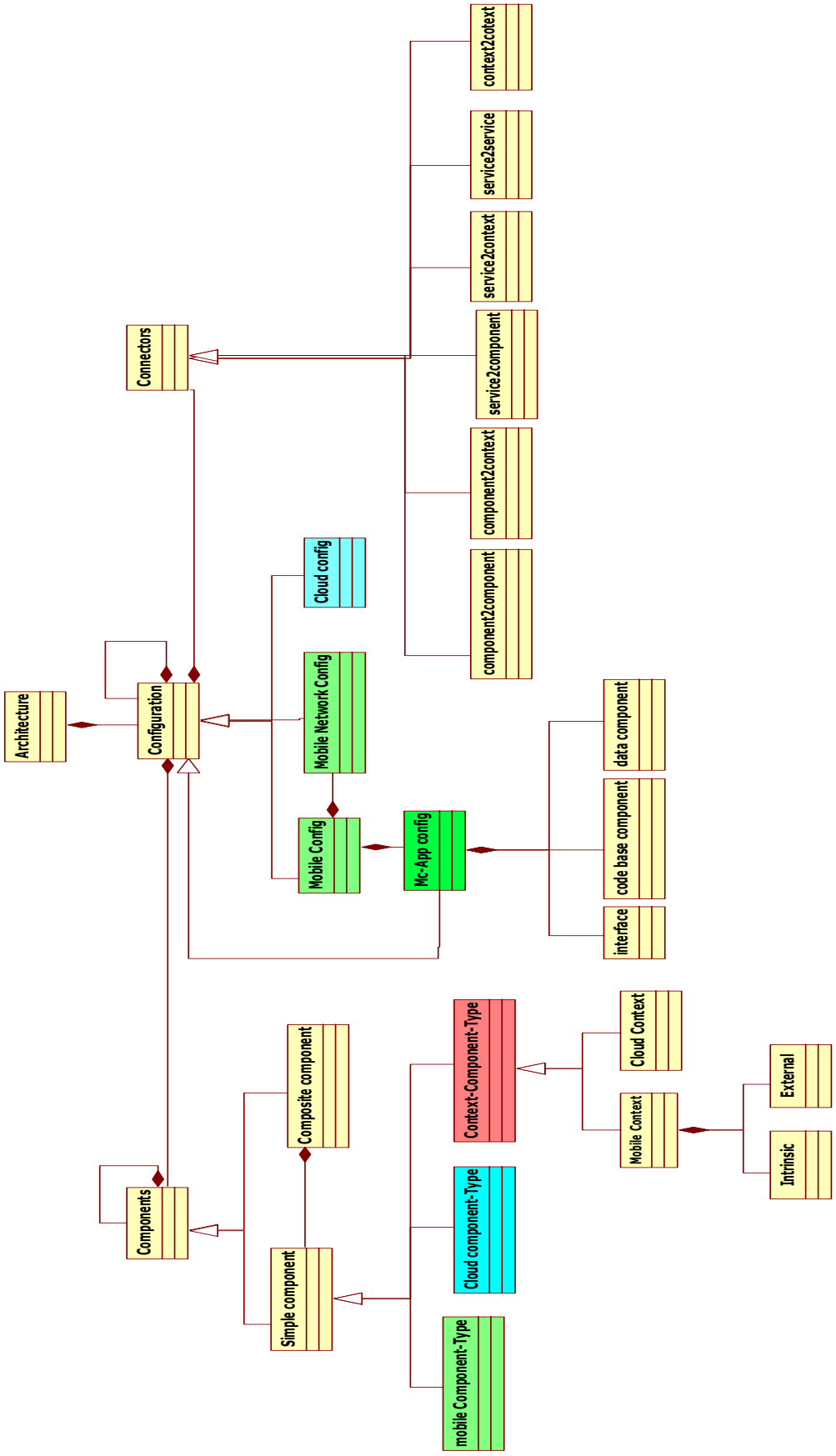


Figure 5. 8 Diagramme de classe présentant le méta modèle de MC-ADL.

5.6.2.3 Le connecteur

La description des interactions entre les composants de l'architecture est faite par le biais des connecteurs. Un connecteur définit l'ensemble des instructions pour décrire les interactions entre les éléments architecturaux ainsi que les règles qui gèrent cette interaction. Les connecteurs sont des entités architecturales qui agissent comme des médiateurs entre les éléments architecturaux. Les connecteurs décrivent aussi des interactions complexes, comme un protocole client/serveur ou un lien SQL entre une base de données et une application (Amirat and Oussalah 2009).

Les connecteurs dans un ADL peuvent regrouper des règles pour implémenter un type spécifique d'un connecteur. Comme souligné dans (Bass 2007) e t(Medvidovic and Taylor 2000), généralement les connecteurs dans les langages de description d'architectures peuvent être classés en trois groupes : les connecteurs implicites, les ensembles énumérés de connecteurs prédéfinis et les connecteurs dont les sémantiques sont définies par les utilisateurs.

Les connecteurs implicites définis par MC-ADL sont ceux qui relisent les composants avec la configuration. Par contre il définitplusieur types de connecteurs explicites reliant différents composants de même ou de différents types (voir le méta-modèle dans la figure 5.8).

5.6.3 Évaluation du MC-ADL

Dans cette section, nous allons évaluer MC-ADL en utilisant: *ADL classification and comparison framework* proposé par Medvidovic et *al*. dans (Medvidovic and Taylor 2000). Une lecture attentive de l'article est recommandée pour une meilleure compréhension de l'évaluation et les critères utilisés pour classer et évaluer MC-ADL (voir table 5.1).

5.6.3.1 Architecture

MC-ADL partage avec les ADLs traditionnels les éléments de base (c.-à-d. composant, connecteurs et configuration) et se diffère d'eux en ajoutant des nouveaux types aux éléments en fonction de leurs environnements de déploiement (c.-à-d. Cloud ou Mobile).

5.6.3.2 Composant

Chaque composant est modélisé séparément d'une spécification de mise en œuvre (à savoir, le composant peut être mis en œuvre en utilisant la technologie de programmation adéquate choisie par le développeur) pour soutenir la réutilisation et la généralisation des concepts. Ainsi, cette hétérogénéité posera de nouveaux défis qui ont été identifiés et traités dans Smart Cloud Gate présenté dans les sections à venir. Les interfaces des composants se varient selon leurs types, notamment : les interfaces *port requis et port fournis* des composants de type mobiles, les interfaces de type « *service required » and « service provided »* des composants Cloud, et enfin les interfaces « *parmaters provided and « parametres required »* des composants « *context* ».

Les interfaces d'un composant expriment les contraintes sur l'interaction entre les composants. L'évolution des composants est traduite sous forme d'un sou-typage des composants types. Les propriétés non fonctionnelles représentent les paramètres de génération, d'évolution et les attribues qui relisent le composant avec son environnement d'exécution.

5.6.3.3 Connecteur

L'interaction entre les différents éléments requiert une nouvelle classe de connecteurs. Ainsi, MC-ADL propose deux types généraux de connecteurs: un type entre les composants hétérogènes et un autre type entre les composants homogènes. Chaque type de connecteurs a de nombreux sous-types qui modélisent l'interaction entre les éléments bien spécifiques. Les contraintes sur les connecteurs sont exprimées via les protocoles de communication. Il est à noter que l'évolution des connecteurs dans MC-ADL est définie sous forme d'ajout de nouveaux connecteurs en appliquant le sou-typage sur des connecteurs types déjà existants.

Features / **MC-ADL Elements**		***Characteristics***	***Interfaces***	***Type***	***Semantics***	***Constraint***	***Evolution***	***Non-Functional Properties***
Component	Mobile Component	Implementation independent	Port; Required/provided port	Extensible type	No support can use other semantic model	Via interfaces	Via Sub typing	Attribute and parameters needed
	Cloud Component	Implementation independent	Service; Required/provided service	Extensible type	No support can use other semantic model	Via interfaces	Via Sub typing	Attribute and parameters needed
	Context Component	Implementation independent	Parameters; Required/provided parameters	Predefined type	Xml model	Via interfaces	Via Sub typing	Attribute and parameter needed
Connectors	Between Heterogeneous components	Connector; explicit	None	Component2componet Component2context	Implicit in connector type	Protocol of interaction	Via Sub typing	None
	Between homogeneous components	Connector; explicit	None	Component2componet Component2context	Implicit in connector type	Protocol of interaction	Via Sub typing	None

Table 5. 1 MC-ADL support for modeling component connectors.

5.6.3.4 Configuration

Un lien de traçabilité est maintenu entre le modèle d'architecture et MC-App au niveau de l'exécution afin d'assurer la cohérence entre les deux niveaux. La configuration de l'architecture prend en charge la modélisation des éléments hétérogènes et fournit un

haut niveau d'abstraction (voir tableau 5.2). La configuration générale représentant l'architecture de l'application regroupe les configurations de chaque environnement (c.-à-d. Configuration Cloud et configuration Mobile). En général la configuration supporte le mécanisme de dynamité en ajoutant, supprimant ou modifiant les composants qui les encapsulent. MC-ADL n'applique pas des contraintes sur la configuration, car l'assemblage des composants et les interactions entre eux subit déjà à des contrôles et des contraintes.

	Characteristics	Understand	Compos	Refine/trace	Heterogeneous	Scalability	Evolution	Dynamism	Constraint	Non-Functional Properties
Mobile and Cloud Reconfigurations	Explicit graphical and textual specification	None	Support through composite component	From Archi to implemented App	Between mobile and cloud elements	Aided by the addition of cloud service and MC-Apps	Supported in the architecture via predefined and dynamic evolution scenarios	Dynamic adding, removal modifying replacing elements	In terms of communication protocol on c	None

Table 5. 2 MC-ADL support for modeling Configuration.

5.6.4 Mise en œuvre de MC-ADL

Le projet *Eclipse Modeling Framework* (EMF) est un outil de modélisation qui permet la génération de code et des applications basées sur des modèles de données structurées. EMF permet aussi de produire des classes Java représentant le modèle avec un ensemble de classes pour adapter les éléments du modèle afin de pouvoir les visualiser, les éditer avec un système de commandes et les manipuler dans un éditeur.

En utilisant EMF, nous avons créé deux types de modèles d'un côté un modèle définissant des concepts où le méta-modèle simplifié de MC-ADL est présenté dans la figure 5.9 et de l'autre côté un modèle instanciant ces concepts. Ce dernier est décrit à l'aide d'une palette (voir figure 5.10) générée en utilisant l'outil emfatic d'Eugenia sous Éclipse (eugenia 2016).

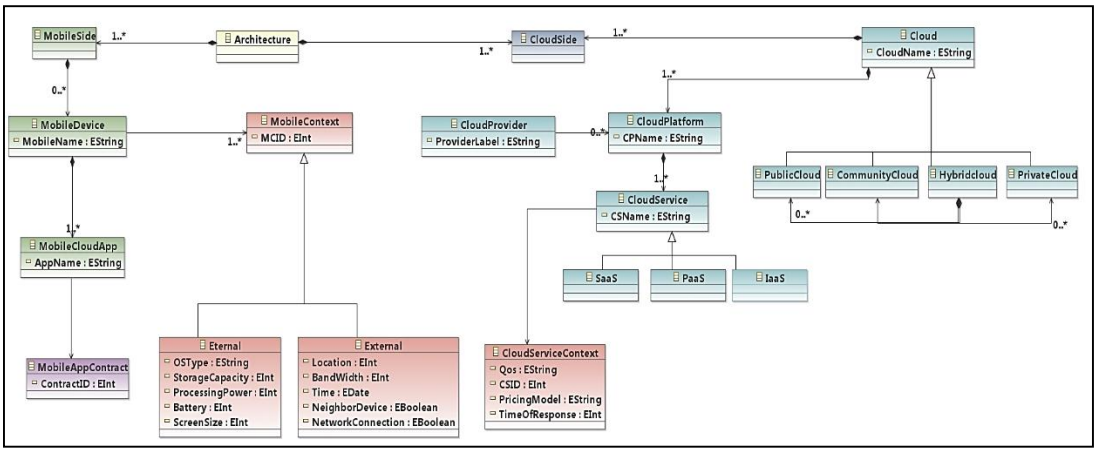


Figure 5. 9 Meta Modèle de MC-ADL sous éclipse.

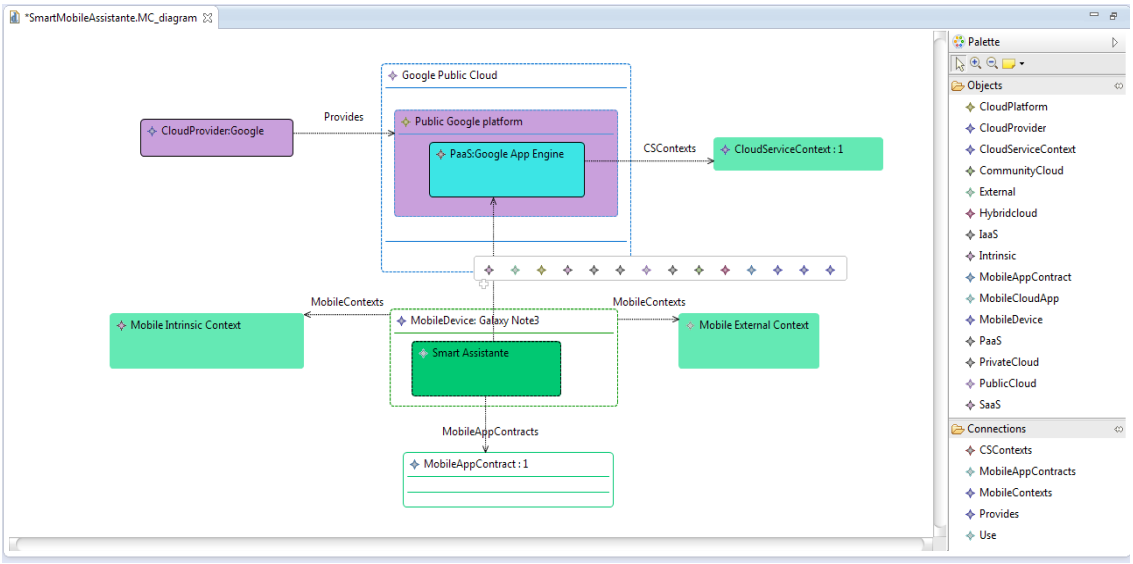


Figure 5. 10 Palette de description de MC-Apps.

5.7 Modélisation des informations contextuelles

Dans le monde à venir de « *Pervasive Computing* », les utilisateurs accèdent au World Wide Web par une grande variété d'appareils mobiles offrant des capacités hétérogènes (en ce qui concerne l'affichage, la saisie des données, la capacité de calcul, etc.). Ces périphériques sont connectés à l'Internet par divers systèmes de communication offrant des fonctionnalités différentes (bande passante, délai, etc.).

La clé pour répondre aux exigences de ce milieu hétérogène est l'adaptation du contenu afin d'assurer une présentation optimale en fonction des capacités des dispositifs et des caractéristiques de la connectivité du réseau. Cependant, nous ne pouvons pas répondre objectivement sur la question de la présentation optimale. Cela dépend plutôt sur les préférences particulières de l'utilisateur. En conséquence, l'adaptation du contenu doit tenir compte des informations contextuelles de l'appareil, la connexion réseau et l'utilisateur.

Afin de permettre une adaptation sensible au contexte, des informations contextuelles doivent être recueillies et éventuellement présentées à l'application pour effectuer l'adaptation. Par conséquent, un format de représentation commune pour les informations de contexte est nécessaire. Plusieurs approches ont été proposées pour modéliser le contexte à savoir : *Key-Value model, Markup Scheme Models, Object Oriented Model, Logic Based Model, Ontology based Model*, et enfin le modèle utilisé dans cette thèse une représentation graphique en utilisant un modèle à base des annotations *Unified Language Modeling* (UML) (Bettini *et al.* 2010).

Dans notre travail, non seulement nous favorisons l'utilisation du contexte, mais aussi nous visons à l'utiliser de façon intelligente pour cela nous avons combiné le concept de *context awareness* (Chen and Kotz 2000) avec la *reflectiveness* (Kon *et al.* 2002). Une application réflective c'est une application ayant la possibilité de raisonner sur le contexte et changer son propre comportement. Pour cela, nous avons introduit le *MC-App Contract* qui encapsule les informations contextuelles selon des conditions établies par le développeur lors de la conception de l'application.

L'association du contrat avec l'application est faite au niveau architectural. Alors, nous avons proposé le méta-modèle présenté dans la figure 5.11 pour modéliser le contrat d'une façon graphique, générique et compatible avec l'architecture de MC-App.

Le contrat se compose des *term* qui modélisent les conditions selon lesquelles les informations sont organisées. Chaque *term* provoque un certain comportement de l'application. Les informations contextuelles sont organisées dans la classe *term* en deux catégories notamment :

Cloud Service Context : Le développeur lors de la conception des MC-Apps définit un ensemble de services avec lesquelles interagissent MC-Apps et les caractéristiques des services à respecter (ex. mode de paiement, temps de réponse et QoS).

Mobile Context: Il est à noter que le contexte mobile dans notre Framework n'est pas limité sur la position et le temps de l'appareil, mais aussi il comprend les caractéristiques intrinsèques de l'appareil (la taille de l'écran, le CPU, RAM, capacité de stockage, pourcentage capacité de la batterie, etc.) et les caractéristiques extrinsèques de l'environnement (bande passante, disponibilité de connexion et les appareils à proximité).

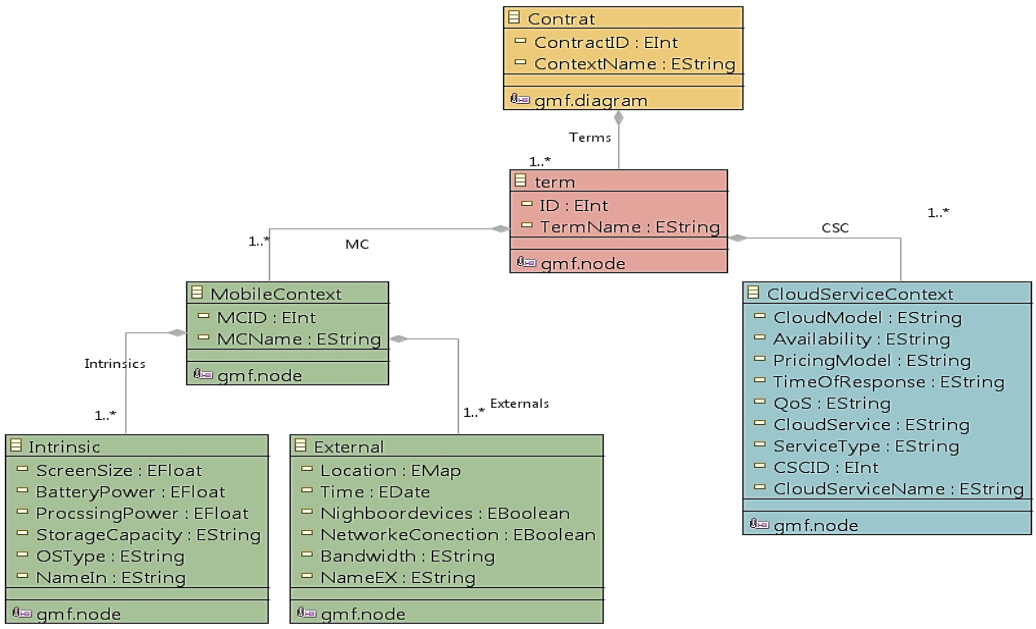


Figure 5. 11 Diagramme de classe représentant le méta modèle du Contrat sous éclipse.

En utilisant les technologies EMF, Emfatic d'Eugenia sous éclipse (Éclipse 2016) la palette qui permet la modélisation du contrat est présentée dans la figure 5.12.

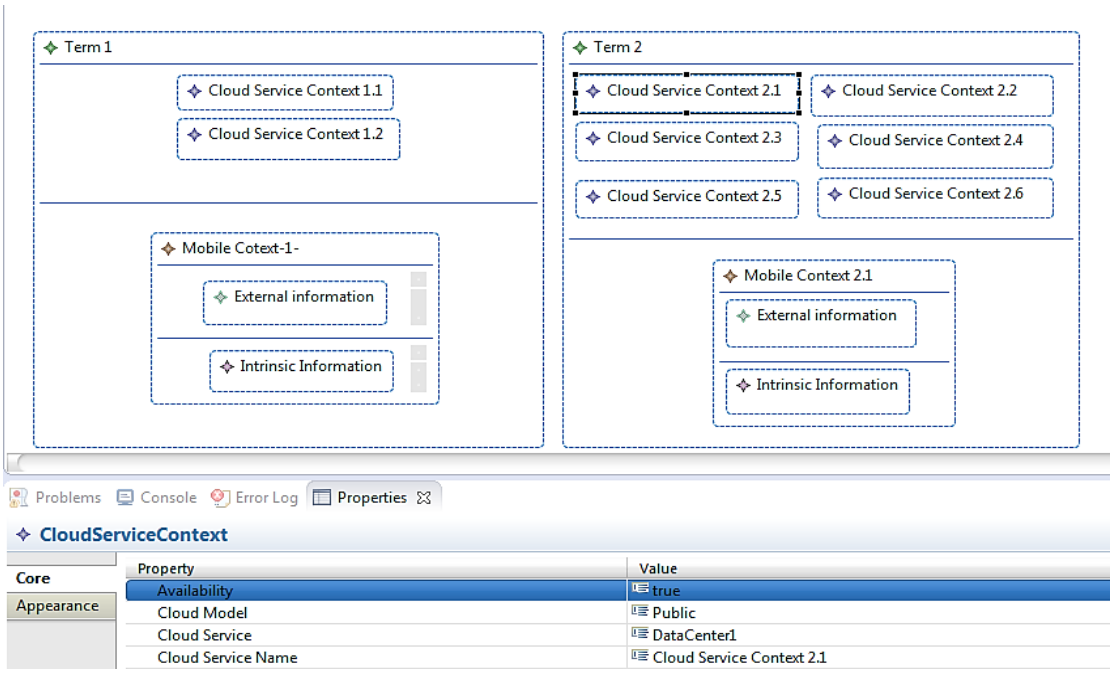


Figure 5. 12 Modèle graphique de contrat de MC-App.

5.8 Architecture de Smart Cloud Gate

Dans le but de prendre avantage du Cloud, nous croyons que la clé est de garder une conscience constante sur le contexte Cloud et mobile simultanément. Cette connaissance permettra aux développeurs de suivre l'actualité au niveau de Cloud et fournir aux applications les services les plus adéquats. Pour favoriser ce concept, nous avons associé un contrat à l'application. Le modèle de contrat comme montré dans la figure 5.12

regroupe toutes les informations contextuelles recueillies depuis le contexte mobile et le contexte Cloud.

Pour réaliser ces visions, nous proposons Smart Cloud Gate dont l'architecture est présentée dans la figure 5.13.

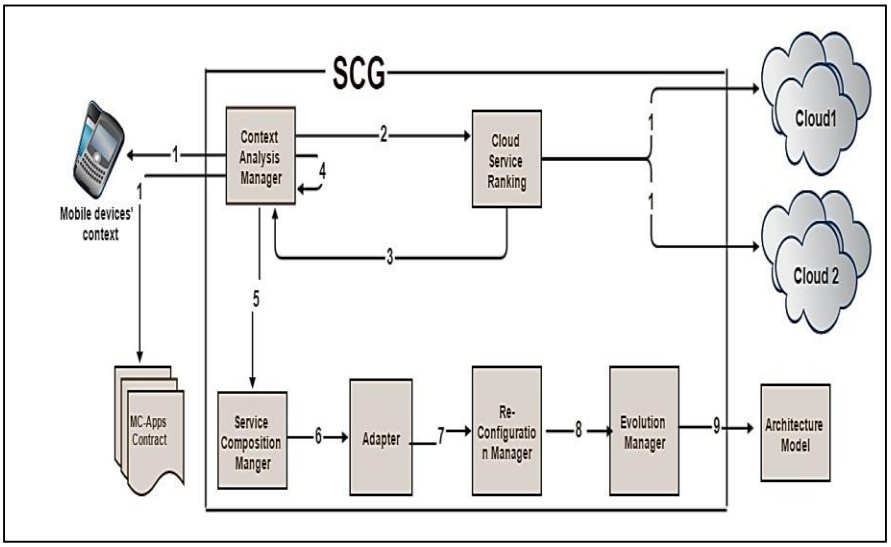


Figure 5. 13 L'architecture de Smart Cloud Gate.

La logique d'interaction des SCG est présentée comme suit :

Le *Context Analysis Manager* (1) extrait les informations du contexte actuel de l'appareil mobile et les règles du contrat de l'application. Simultanément, *Cloud Service Ranking* (1) profile les services Cloud et applique une mise à jour de base de données de services. Après la recueille des informations contextuelles, le *contexte Analysis Manager* (2) demande du *Cloud Service Ranking* l'ensemble recommandé de Service Cloud. Ce dernier envoie l'ensemble le plus actuel (3). Le *Context Analysis Manger* étudie la correspondance entre l'ensemble actuel des services, celui demandé par l'application et le contexte courant. Cette étude peut résulter deux scénarios : Le premier scénario, si n'y aura pas de correspondance entre l'ensemble courant et celui indiqué dans les termes du Contrat dans ce cas un nouveau *Contract* sera créé. Le deuxième scénario, c'est le cas contraire où il y aura une correspondance (5). En se basant sur le deuxième scénario et après avoir vérifié la correspondance, l'ensemble de services sera envoyé au *Composition Manger.* Un seul service ne peut pas satisfaire les exigences des MC-Apps, par contre en utilisant plusieurs services fournis par différents fournisseurs et implémentés dans différentes plateformes avec des différents APIs posent des nouveaux problèmes et défis pour le développeur. Pour rependre à cette problématique, nous proposons *Cloud Composition Manager* qui englobe les services demandés dans un seul service abstrait utilisable par l'application. L'adaptation de ce service composite est faite par *Adapter* qui lui ajoute une interface adaptable avec MC-Apps (6). Enfin, un ordre de reconfiguration est envoyé au *Reconfiguration Manger* (7) pour exécuter une reconfiguration dynamique de l'application et ensuite invoquer une évolution architecturale à travers la provocation d'*Evolution Manger* (8).

5.8.1 Traitement des informations contextuelles

Le processus de traitement des informations contextuelles commence par l'acquisition des informations contexte jusqu'à leurs utilisations à travers la couche d'application.

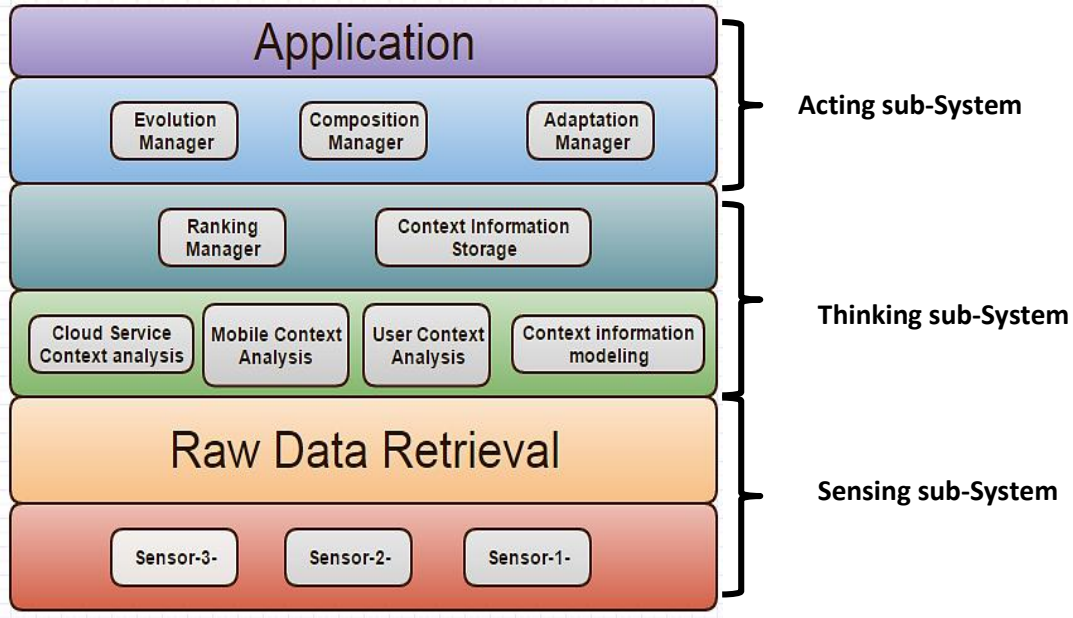


Figure 5. 14 Architecture multi couches de Smart Cloud Gate.

Une architecture en couches (figure 5.14) fournit des avantages considérables pour le développement d'applications sensibles au contexte. Certains avantages qui peuvent être garantis à partir d'une approche en couches comprennent: l'augmentation de l'espérance de vie, la mobilité, la modularité, l'interopérabilité, une plus grande compatibilité et une meilleure flexibilité.

5.8.1.1 Acquisition des informations contextuelles

Les informations contextuelles dans cette thèse concernent également le Cloud, le Mobile et l'utilisateur. Par conséquent, l'acquisition des informations contextuelles est faite en trois étapes :

Etape1 : En utilisant les entités de sensation de l'appareil « *Sensor* », des informations « *extrinsèques* » comme : la localisation, le temps, la température et les appareils à proximité de l'utilisateur sont collectés.

Étape 2 : Les informations de l'appareil mobile « *Intrinsèques* » comme : la capacité de stockage, la puissance de traitement, la RAM et l'état de la batterie sont déjà définies au niveau architectural dans le modèle préliminaire de contrat, mais elles peuvent être mises à jour en surveillant l'état de l'appareil lors de l'exécution de l'application. Par exemple la diminution de capacité de stockage.

Étape 3 : Cette étape concerne la collection des informations sur les services Cloud. Ces informations seront classifiées selon plusieurs critères (ex. prix, capacité, disponibilité, etc.) afin de faciliter la sélection du service le plus adéquat.

L'extraction des données brutes représentant les informations contextuelles est faite par le bai des APIs ou des drivers qui jouent les rôles des interfaces de « *Sensor* ».

5.8.1.2 Traitement des informations contextuelles

Les informations contextuelles des mobiles et de Cloud seront organisées selon des critères de classification pour faciliter le raisonnement sur eux.

Une seule collection d'informations contextuelles de Cloud est définie dans cette thèse comme le *CloudView.* Ce dernier représente une perspective sur le Cloud dans un état donné. La modélisation des informations contextuelles est faite en définissant les modèles *Contract*. Non seulement le modèle de contrat donne une représentation compréhensible des informations contextuelles, mais aussi il définit un mécanisme de raisonnement expliqué comme suite :

Un terme dans le contrat organise les informations contextuelles du mobile et du Cloud. Cette organisation n'est pas faite aléatoirement, mais plutôt pour définir la situation qui gouverne un comportement donné de l'application.

Comme expliqué dans la figure 5.15, un comportement X peut être provoqué, si dans un état donné le contexte mobile est dans l'état Y et le contexte Cloud est dans l'état Z ou Z+1.

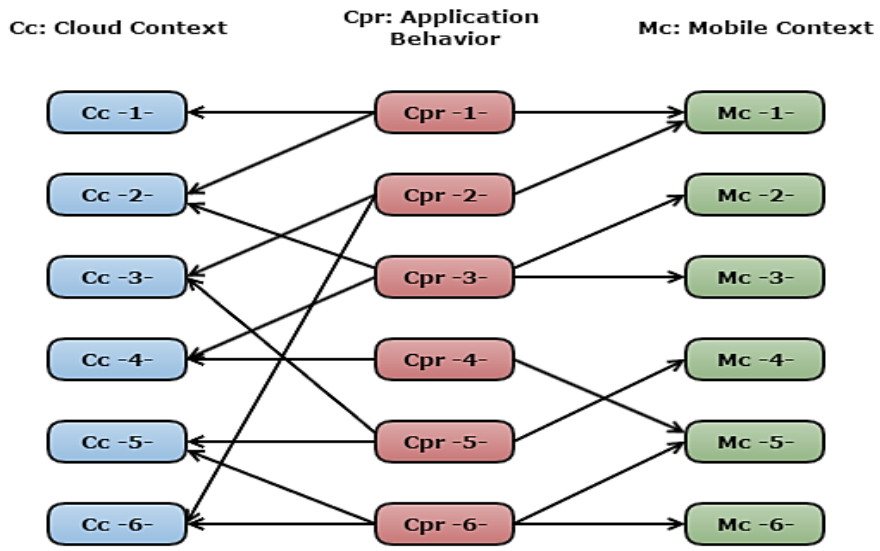


Figure 5. 15 Le raisonnement sur les informations contextuelles.

5.8.1.3 Exploitation des informations contextuelles.

Nous avons attesté précédemment que les informations contextuelles jouent un rôle très important dans le développement des applications mobiles à base de Cloud.

L'objectif principal de cette thèse est de proposer une méthode effective et intelligente pour modéliser des MC-Apps performantes en fonction d'exploitation intelligente du Cloud et du contexte mobile. Pour réaliser cet objectif, nous avons introduit les informations contextuelles dans le niveau architectural ainsi que nous avons proposé un mécanisme afin de les exploiter intelligemment en utilisant Smart Cloud Gate. Le

diagramme de séquence présenté dans la figure 5.17, illustre comment les informations contextuelles sont utilisées pour améliorer la performance des MC-Apps par le biais d'une reconfiguration dynamique et une évolution architecturale anticipée et non anticipée.

a) Évolution architecturale

L'architecture logicielle n'est pas conçue d'une façon à rester stable tout au long la durée de vie de l'application. Comme l'application subit des circonstances qui engendrent un changement dans sa structure (c.-à-d. reconfiguration dynamique), ces dernières doivent être également présentes au niveau de son architecture. Afin de garder la cohérence entre l'application et son architecture, une évolution de cette dernière est indispensable. Nous pouvons définir l'évolution d'architecture logicielle comme : les différents changements qui touchent sa structure et ces éléments constitutifs. Les changements engendrés sont le résultat d'exécution des opérations d'évolution telles que la modification, l'ajout ou la suppression.

MC-ADL implémente une évolution architecturale dynamique (c.-à-d. au cours d'exécution de l'application) anticipée (c.-à-d. les changements à appliquer, selon une situation donnée sont déjà prévus). Le mécanisme d'évolution est sous forme de transformation de modèles comme exprimé dans la figure 5.16. L'architecture change de version$_i$ à une version$_{i+1}$ en appliquant des règles de transformation qui traduisent une opération d'évolution. Le mécanisme de transformation est implémenté en utilisant Atlas Transformation Language (Jouault *et al.* 2006) (Language 2016).

L'évolution architecturale est la conséquence d'une reconfiguration dynamique de MC-App qui est à son tour le résultat de changement de contexte. Le modèle de ce dernier servira comme un deuxième modèle qui reflète les paramètres d'évolution dans le processus de transformation.

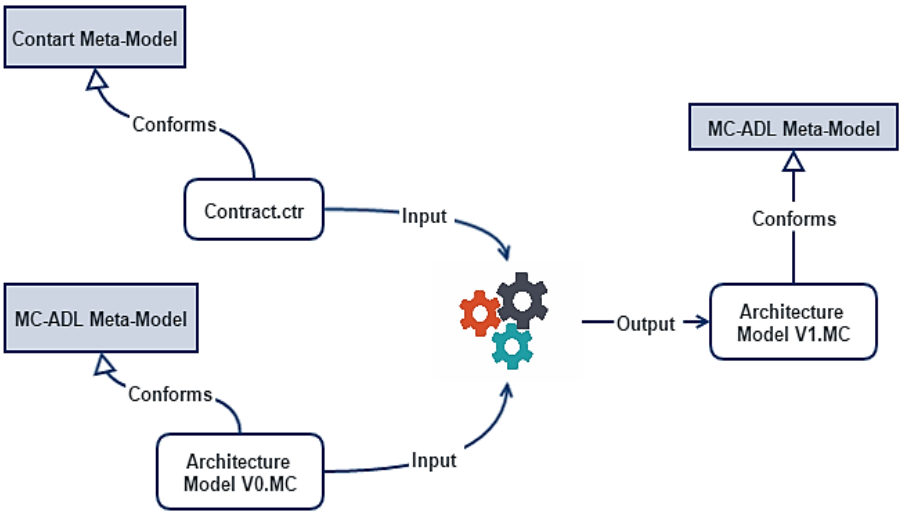


Figure 5. 16 Mécanisme d'évolution architecturale.

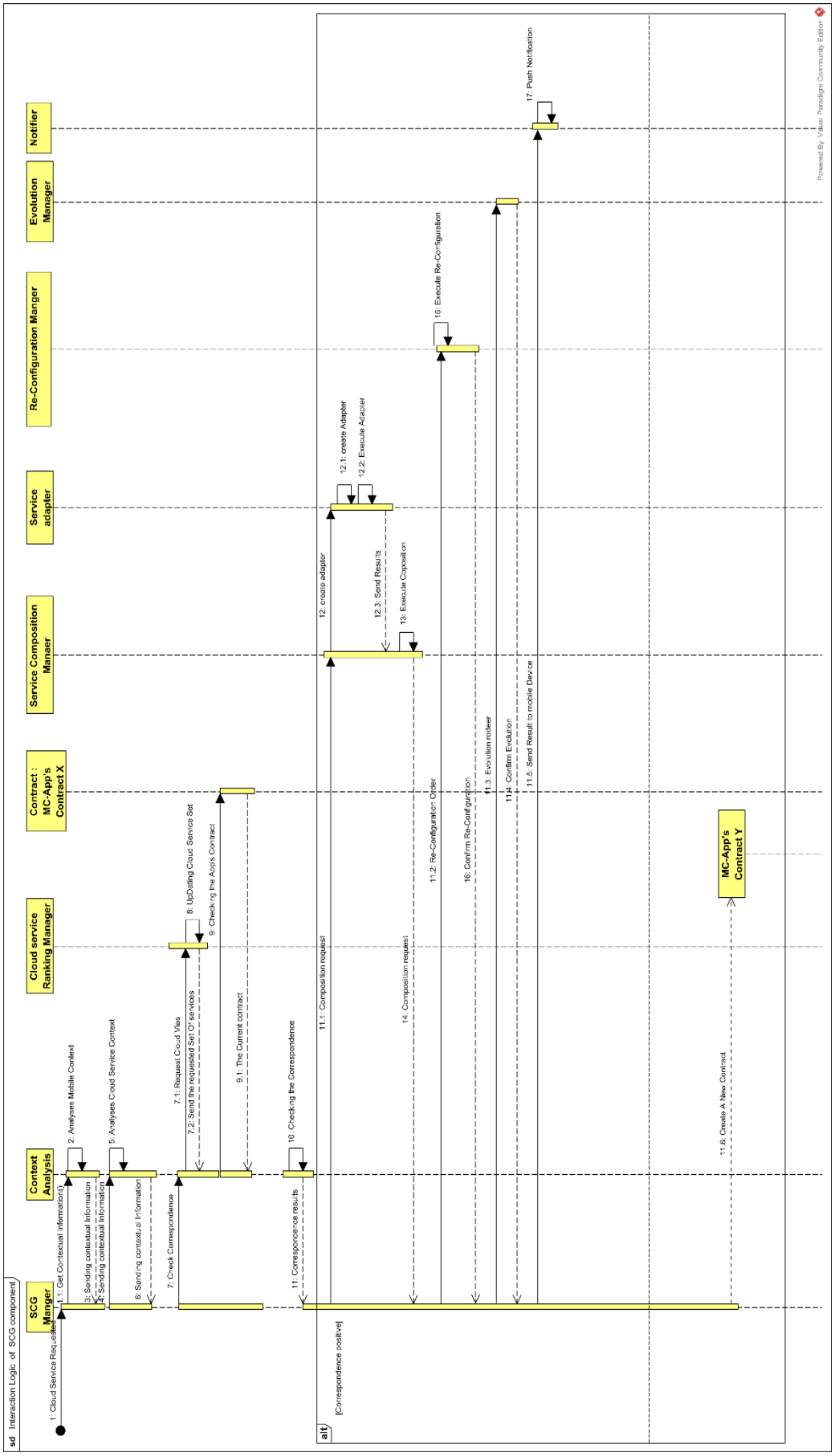


Figure 5. 17 Diagramme de séquence illustrant l'exploitation des informations contextuelles.

L'évolution architecturale en termes d'opérations (ex. la suppression d'un service), nécessite l'exécution de plusieurs règles ATL selon un ordre bien défini. Ce dernier est exprimé dans une classe Java nommée « *Evolution startegy* » (voir listing 5.1). ATL utilise des requêtes sous forme d'expressions OCL (Warmer and Kleppe 1998), mais dans notre Framework afin de garder la cohérence nous utilisons la version Java des règles ATL.

Algorithme 1 : Delete Service Strategy

```
Var: String: Service, Platform, Provider;
Delete_Service_Strategy (Service) {
    If Service_Availability (ServiceName)==True; and Services_Nbr()=1; then
        If Service_Platform (Platform)=true then
            If Service_Provider (Provider)==true; then
                Delete_Provider_Strategy (Provider);
                Delete_Platform_Strategy (platform);
                Delete_Service_ATL_Rule(Service);
            Else error_msg("the variable Provider does not exist");
        Else error_msg("the variable platform does not exist");
    Else {
            if Service_Nbr()!=1 then
                Delete_Service_ATL_Rule(Service) ;
            Else error_msg("Service is no longer") }
End;
```

Listing 5. 1 La stratégie de suppression d'un service.

Un cas de suppression d'un service déjà au cours d'utilisation est illustré dans le listing 5.1. Une seule plateforme peut offrir plusieurs services, mais une seule plateforme est fournie par un seul fournisseur. Alors, avant de supprimer un service, il est indispensable de vérifier s'il n'est pas le seul service fourni par la plateforme existante. Si c'est le cas, la suppression de service engendra la suppression de plateforme et le fournisseur. Le même principe s'applique pour la suppression de chaque élément dans l'architecture afin de garder la cohérence interne de l'architecture.

b) La reconfiguration dynamique

La reconfiguration dynamique d'un système reflète les changements dans sa structure au cours d'exécution. Dans ce travail de recherche, nous définissions deux types de reconfiguration :

– *Reconfiguration interne* : représente les changements dans la structure interne de MC-App en ajoutant, supprimant, modifiant les composants constitutifs de l'application (Code de base, les données et les interfaces).

– *Reconfiguration externe :* représente les changements dans l'environnement externe de l'application. Nous nous intéressons que par l'environnement Cloud qui change en termes d'ajout d'un nouveau service, la suppression d'un service déjà au cours d'utilisation par l'application ou le replacement d'un service utilisé par un autre plus adéquat.

5.9 Mobile Cloud Simulation Toolkit(MC-SIM)

Nous attestons qu'un ADL doit offrir des moyens pour définir des architectures dans le but de produire des applications efficaces et performantes, mais aussi fournir un moyen afin de les tester avant de les mettre en œuvre. Par conséquent, une simulation du comportement de l'application mobile dans un environnement Cloud est préalable et indispensable avant de se diriger vers le déploiement réel. À cette phase, le développeur peut tester son application dans un environnement simulé, gratuit, contrôlable et assurer la mise en œuvre d'une application sans future panne/bug.

Après avoir étudié plusieurs outils de simulation (cf. chapitre 4) nous avons conclu deux points :

1- Il n'existe pas d'approches ou des suggestions d'étudier et d'analyser le comportement des applications mobiles dans l'environnement Cloud

2- CloudSim est le simulateur le plus simple et adéquat pour une extension potentielle afin de répondre au manque de simulation des applications mobiles à base de Cloud.

CloudSim ne propose pas des classes pour simuler les entités mobiles (ex. appareil mobile, application mobile et réseau mobile) car il n'est pas développé pour les MC-Apps. Pour répondre à cette limite, nous avons ajouté de nouvelles entités de simulation au CloudSim dédié à la simulation des MC-Apps. Ces entités sont implémentées en héritant les fonctionnalités de base de CloudSim.

La figure 5.18 montre l'architecture en couches MC-Sim basée sur CloudSim. « MC-App Cloudlet » est une classe qui hérite du Cloudlet et encapsule toutes les relations entre la tâche d'application et le service de Cloud utilisé. « Mobile Device » encapsule les informations concernant le dispositif mobile (ex. caractéristique intrinsèque et extrinsèque).

En plus des entités déjà existantes de CloudSim (cf. section 4.3.1.2), les entités ajoutées sont présentées dans le diagramme de classe réduit du MC-Sim (voir figure 5.19).

MC-App : Une classe abstraite représentant l'application mobile à base de Cloud et définit le nombre des composants constituant l'application (c.-à-d. Composant de base, composant de donnée)

MC-App CoudLet : Représente les composants exécutables ou bien ceux des données de l'application. Cette classe modélise les mêmes fonctionnalités attribuées par CloudSim à *CloudLet* (cf. Section 4.3.1.2 (b)).

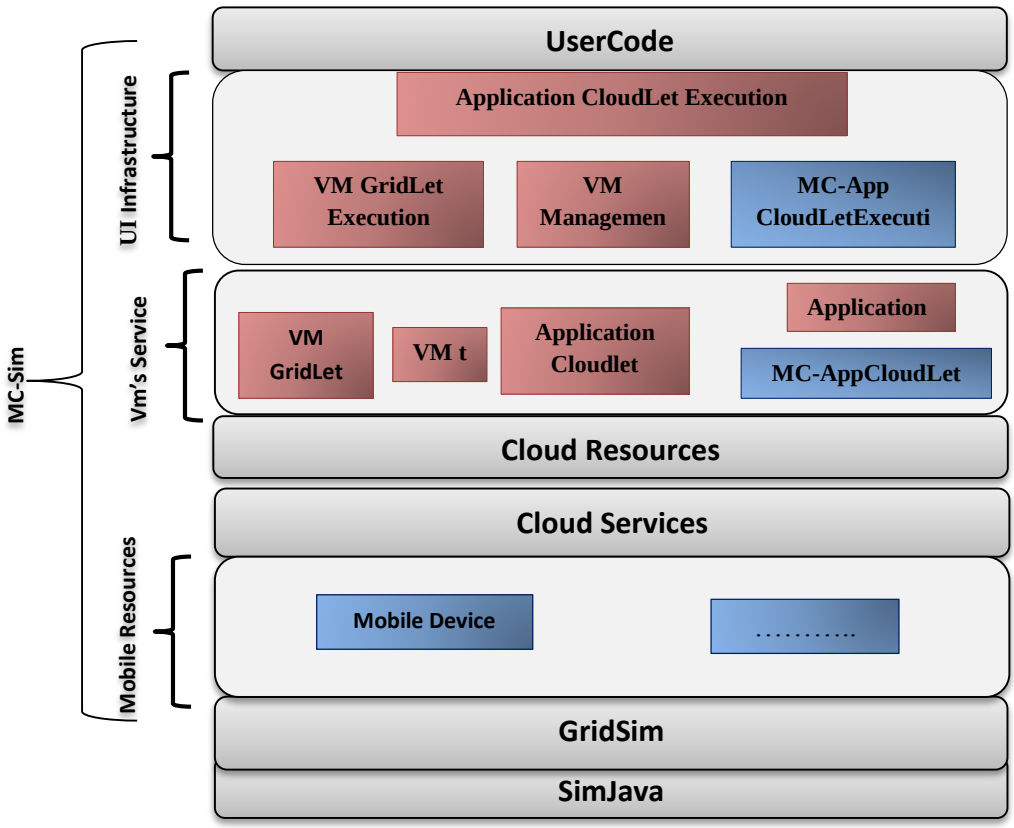


Figure 5. 18 Architecture de MC-Sim.

MC-App CloudLetList : Une classe qui modélise la liste des MC-App CloudLet. Le nombre de MC-App CloudLet correspond au nombre de composants de l'application définie au niveau architectural.

MobileDevice : Cette Classe est générée automatiquement depuis le modèle architectural. Elle encapsule les caractéristiques du mobile (c.-à-d. Contexte intrinsèque).

MobileSensors : Cette interface a le rôle d'instanciation des capteurs mobile comme (Location sensors, time sensors, temperature sensors). Cette interface hérite les fonctionnalités de base de la classe « *Sensors* » de CloudSim en ajoutant de nouveaux paramètres utilisables par la classe « *Scénario* ».

Il est à noter que, notre architecture favorise l'utilisation des services Cloud de différents niveaux (SaaS, PaaS, et IaaS).Par contre dans la phase de simulation, en choisissons CloudSim comme outils à étendre, nous nous sommes limités au niveau IaaS. En outre, MC-ADL favorise la composition hétérogène de l'application en termes de composants exécutables en Cloud ; localement ou en dépendant du contexte. Par contre dans la phase de simulation, nous étudions le comportement de MC-App en exécutant tous les composants au niveau de Cloud.

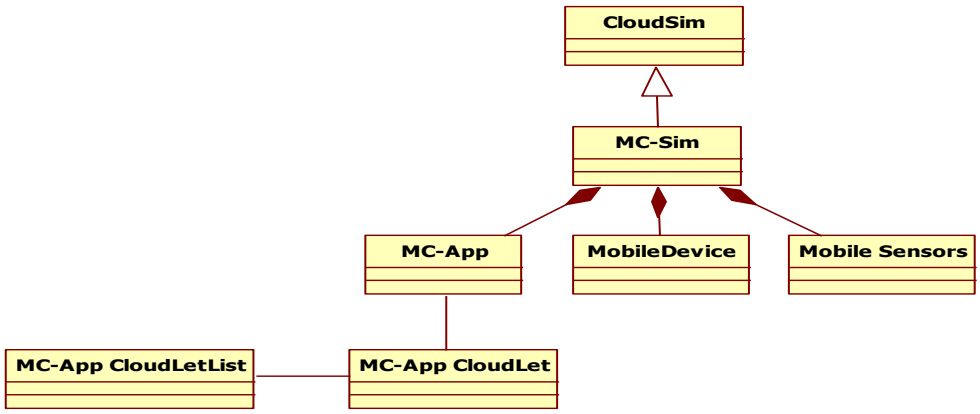


Figure 5. 19 Diagramme de Classe représentant MC-Sim.

Les classes de MC-Sim sont distribuées à travers les packages définis par l'outil CloudSim. Comme présenté dans la figure 5.20. Les classes *MC-App*, *Mobile Device*, *MC-App CoudLet* sont implémentées dans le package « *org.cloudbus.cloudsim ».*

Le package *« org.cloudbus.cloudsim.lists »* définit les listes des entités de CloudSim y compris la liste *« MC-App CloudLet List».*

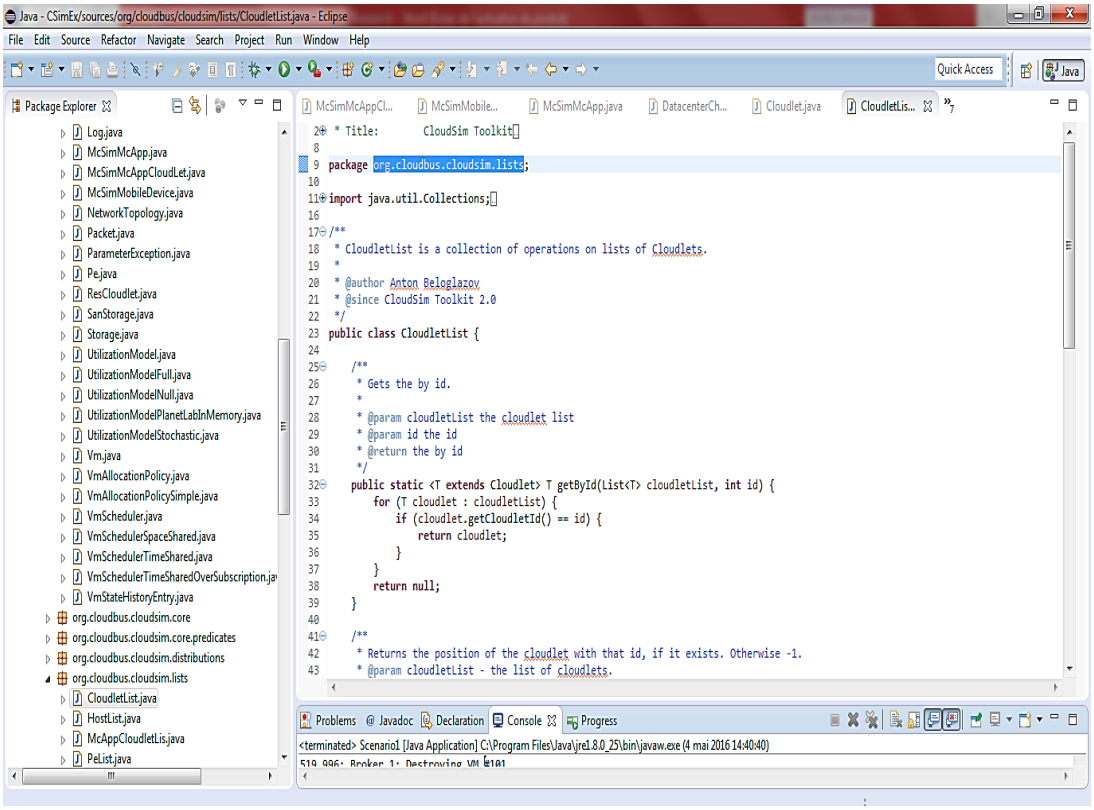


Figure 5. 20 Packages des Simulations.

5.10 Expérimentations

Dans l'objectif d'évaluer le mécanisme d'exploitation des informations contextuelles, que nous proposons on fonction de Smart Cloud Gate. Nous présentons dans cette section

une étude de cas pour étudier le comportement de MC-App selon les changements définis au niveau contexte du Cloud.

5.10.1 Shopping Assistant: Smart Mobile Cloud Application

Shopping Assistant, est une application qui permet de visualiser les magasins à proximité et les visiter virtuellement dans l'appareil mobile sans entrer réellement au magasin. Cette application offre la possibilité d'avoir la liste de tous les magasins à proximité, les chemins à suivre pour les visiter, et garder les articles préférés pour les consulter.

L'architecture de Shopping assistant définis par MC-ADL dans la figure 5.21, illustre que l'application est déployée dans un appareil de type Galaxy edge7 (Choix du concepteur), dont la mémoire interne est de 32Go, et RAM de 4Go (c.-à-d. Informations contextuelles intrinsèques de l'appareil mobile). L'application propose plusieurs fonctionnalités, ou dans notre étude de cas sont exécutées au niveau du Cloud en utilisant les services (de type IaaS) des VMs alloués dans le *Data Center1*. Il est à noter que le service utilisé est fourni dans une plateforme Cloud de type publique.

MC-ADL vise à donner une perspective complète sur l'application et son environnement Cloud et mobile. Nous remarquons que dans l'architecture proposée les détails de Cloud sont définis au niveau de Contexte de Cloud *Data Center Characterstics*, mais plus de détails sur l'application et le Cloud serons donnés lors de la simulation, vu que certains concepts ne sont pas pris en considération lors de la conception de l'application pour renforcer la généricité des entités réutilisables. Par conséquent, la séparation de l'architecture depuis les détails d'implémentation est nécessaire.

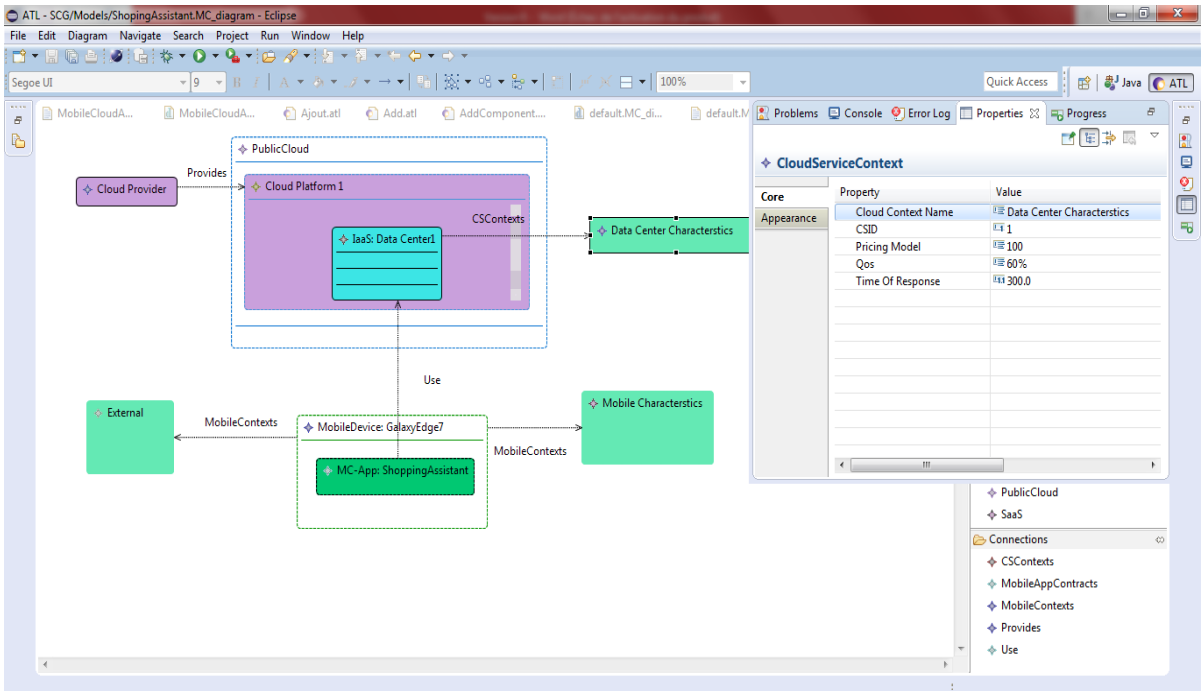


Figure 5. 21 Architecture de Shopping Assistant.

La mise en œuvre de *Shopping Assistant*, est réalisée comme un prototype sous java afin de tester son comportement avec CloudSim. Le code proposé peut être déployé dans l'environnement Android très facilement. Les fonctionnalités du *Shopping Assistant* sont décrites sous forme des classes d'un projet java présenté dans le tableau 5.3.

Classes	Fonctionnalité
User-Account	Une class pour collecter, organiser les informations de l'utilisateur (Id, name, email address, etc.), et enfin les sauvegarder dans des fichiers CSV.
Check-In	Une classe pour identifier l'utilisateur et donner accès à son compte, aussi elle garde les préférences de chaque utilisateur pour les exploiter par la classe recommandation.
Stores	Une classe pour représenter les magasins et sauvegarder leurs informations (id. name, Address, location).
Article	Une Classe qui définit l'ensemble des articles disponibles dans un magasin, les informations à identifier sont : URL, ID, Image, Description Price et disponibilité.
Wish-List	Une classe qui organise les articles choisis par l'utilisateur ainsi que les informations des magasins qui les offrent.
Up-Date-Data	Une classe qui permet la mise à jour de basse de donnée en fonction de changement de location et l'ajout des nouveaux magasins.
Main-Class	La classe principale qui implémente les fonctions de recherche de magasin, l'affichage de localisation et de chemin, et aussi l'affichage des articles disponibles

Table 5. 3 Fonctionnalités de Shopping Assistant en termes de classes Java.

5.10.2 Étude de cas : Étude de comportement de Shopping Assistant dans un changement dans le Cloud

Une des plus grandes chutes de nombreuses applications mobiles est leurs temps de réponse réelle ou perçu. L'importance d'optimiser l'expérience soit donner à l'utilisateur les données qu'ils veulent tout de suite ou d'une certaine manière les distraire de retenir leurs attentions plus longtemps. Les utilisateurs mobiles ces jours sont avertis et attendent les résultats dans une seconde. Le temps de réponse attendu d'une application ne doit pas dépasser une seconde, par contre avec le temps de réponse d'une application à base de Cloud dépend non seulement de l'application, mais aussi de taux d'erreur et la latence du service Cloud.

L'un des objectifs principaux de l'entrelacement du Mobile Computing avec le paradigme du Cloud Computing été de répondre aux problèmes liés aux performances des applications mobiles. Par conséquent, avec chaque solution proposée manifestent plusieurs questions. Par exemple, la délégation de fonctionnalités mobiles au Cloud améliorera la performance de l'application en termes d'énergie, mais infectera le temps de réponse comme expliqué précédemment.

Le but de cette étude de cas est prouvé notre hypothèse que :

« Le Cloud doit être exploité intelligemment afin d'améliorer la performance des applications mobiles sans avoir un effet négatif et ouvrir la porte a d'autre concernes».

Nous allons étudier la performance de l'application en utilisant un service Cloud prédéfini et dans autre cas, étudier son comportement en stimulant sa conscience au contexte, par introduire un changement dans le contexte Cloud. Le deuxième cas est dans le but d'étudier comment l'application tira parti de ce changement pour optimiser son temps de réponse.

L'application comme présentée dans son architecture (voir figure5.21), explique son état initial où elle utilise un seul service Cloud « Data Center1 » auquel elle délègue toutes ces fonctionnalités. La structure interne du Data Center et qui conforme à la description de CloudSim est donnée dans la figure 5.22.

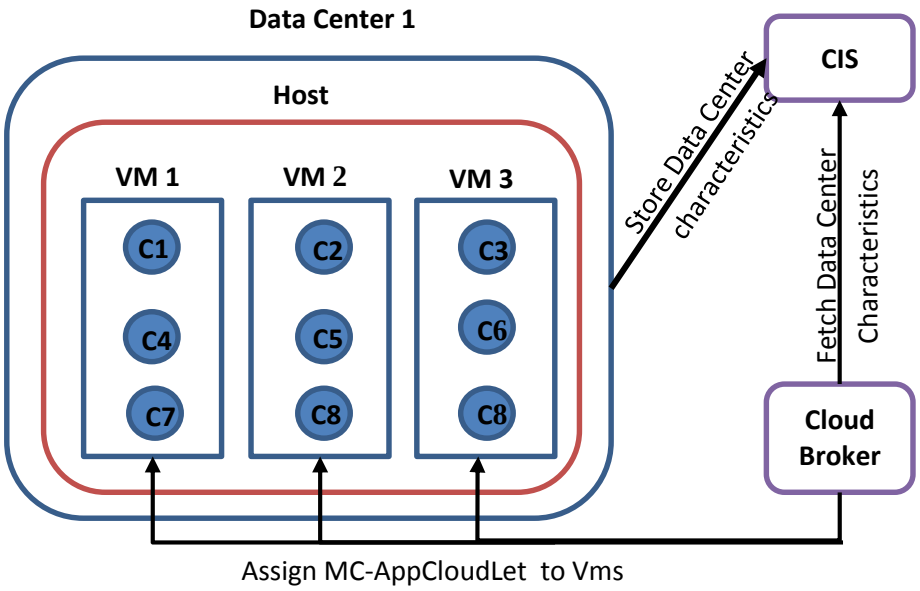


Figure 5. 22 Data Center Structure.

Le Data Center utilisé se compose d'un seul hôte qui héberge trois machines virtuelles auxquelles sont assignées trois MC-AppCloudLet. Les caractéristiques du Data Center (voire listing 5.2) sont sauvegardées dans Cloud Information Service(CIS).

```
// 5. Create a DatacenterCharacteristics object that stores the
//    properties of a data center: architecture, OS, list of
//    Machines, allocation policy: time- or space-shared, time zone
//    and its price (G$/Pe time unit).
    String arch = « x86 »;           // system architecture
    String os = « Linux »;           // operating system
    String vmm = "Xen";
    double time_zone = 10.0;       // time zone this resource located
    double cost = 95;              // the cost of using processing in this resource in dollars
    double costPerMem = 80;        // the cost of using memory in this resource in dollars
    double costPerStorage = 150;   // the cost of using storage in this resource in dollars
    double costPerBw = 0.1;        // the cost of using bw in this resource in dollars
```

Listing 5. 2 Caractéristiques de Data Center

CloudBroker, affecte chaque *MC-AppCloudLet* aux machines virtuelles disponibles, après avoir vérifié quelques contraintes d'assignement (cf. Section 4.3.1.2). Chaque *MC-AppClouLet* est responsable de l'exécution d'une fonction de *Shopping Assistant* comme expliquée dans la table 5.4.

MC-AppCloudLet	Rôle
C1	Sauvegarder les données des magasins.
C2	Sauvegarder les données des utilisateurs.
C3	Sauvegarde les données des articles.
C4	Classifier les articles selon le prix.
C5	Classifier les magasins selon le plus proche au plus loin de l'utilisateur dans sa position actuelle.
C6	Mettre à jour les bases de données.
C7	La recherche des magasins les plus proches.
C8	Établir la liste de préférences
C9	Afficher les magasins les plus proches et le chemin à suivre pour les visiter

Table 5. 4 Fonctionnalités des MC-AppCloudLet.

Dans le contrat associé à l'application existe un terme « terme3.1 » qui indique le partage d'exécution sur plusieurs services (Data Centers) s'ils existent. Alors, l'exécution de scénario'1' résulte l'ajout dynamique d'un nouveau DataCenter'2' (voir figure5.23), ce qui provoque l'activation de la classe « *StartArcitecturalEvolution* » dans le package « *Smart Cloud Gate* ».

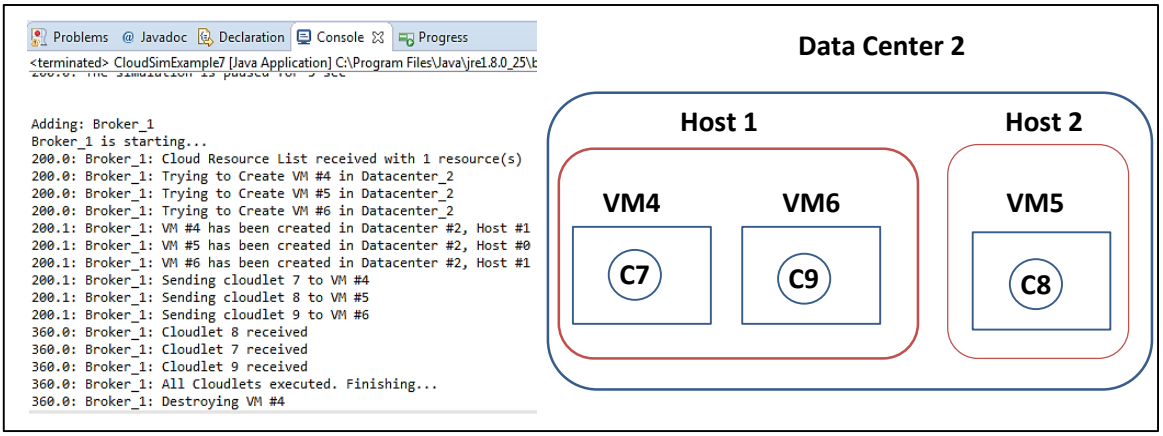


Figure 5. 23 Scénario de simulation.

Le package « *Smart Cloud Gate* » contient des classes Java qui établissent une relation entre le niveau simulation et le niveau architecturale. L'ordre d'exécution et le rôle de chaque classe sont illustrés sont expliqués comme suite.

La classe *Contract-Reader* fais le parcoure du fichier XML représentant le modèle contrat pour extraire les informations contextuelles, ensuite elle fait appel à la fonction *Study-correspondance* qui a comme paramètre l'id de contrat et l'id de scénario. Le résultat de cette fonction est un ensemble de paramètres qui seront envoyés à la classe *Evol-Param* pour établir le modèle de paramètre qui entre comme un deuxième modèle dans l'exécution. Le modèle paramètre est une version réduite du modèle contrat qui comprend les informations nécessaires pour l'évolution architecturale. Finalement, la classe Architecture-Evolution exécute la stratégie d'exécution qui correspond à l'ajout d'un service. Dans ce cas le service est fourni par le même fournisseur dans la même plateforme. (Voir figure 5.24).

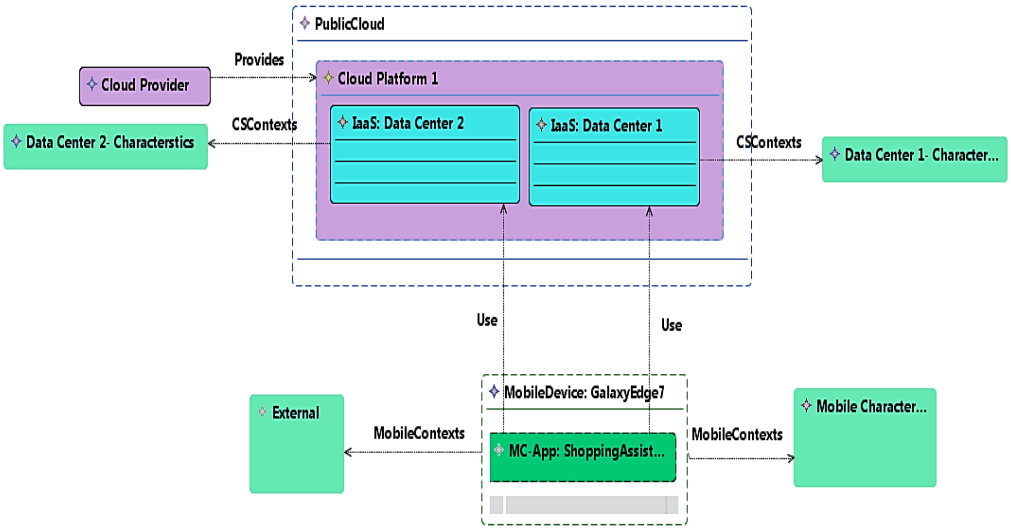


Figure 5. 24 Nouvelle version de l'architecture de Shopping Assistant.

5.10.3 Discussion des résultats

Le terme 3.1 dans le *MC-App Contract* indique l'utilisation des services disponible pour partager l'exécution des composants de l'application. Le but de définir ce terme est,

d'optimiser le temps de réponse afin d'améliorer la performance de Shopping Assistant. Nous avons attesté que le scénario'1' favorise l'allocation des VMs en suivant « *TimeSharedAllocationPlolicies* », où le temps attribué à chaque *MC-App CloudLet* est en fonction de nombre d'instruction (chaque instruction est estimée de prendre une seule milliseconde d'exécution), mais l'ordre d'exécution est organisé successivement. En ajoutant un nouveau Data Center, des nouveaux paramètres d'allocation sont ajoutés ce qui permet l'exécution de Quelques Cloudlet en même temps afin de gagner du temps.

L'état 0 de l'application correspond à l'exécution successive des Cloudlet qui prennent 673ms comme temps total d'exécution (c.-à-d. temps de réponse). Par contre l'état 1 dans le quels les MC-Cloudlet (C1-C7, C2-C8, C6-C9) s'exécute simultanément, a résulter une diminution remarquable a 265ms dans le temps de réponse.

Nous remarquons d'après les résultats obtenus dans la figure 5.25 que l'ajout d'un nouveau service a amélioré la performance de l'application considérablement, ce qu'affirme que l'utilisation intelligente du Cloud en fonction de modèle contrat a été bénéfique. Car, sans le modèle contrat qui modélise une sorte de conscience sur le Cloud, l'application continuera d'utiliser un seul et le même service, en éliminant l'occasion d'exploiter la nouveauté qui peut survenir au niveau du Cloud. Il est à noter que, les résultats obtenus soutiennent nos arguments en proposant *« Context aware Framework for Modeling and Simulating Mobile Cloud Application »*.

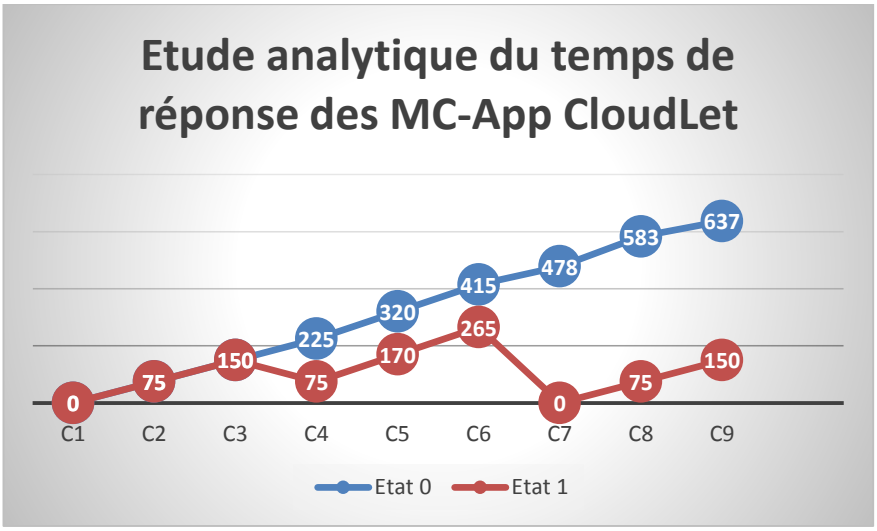


Figure 5. 25 Étude analytique du temps de réponse de MC-App CloudLets.

5.11 Conclusion

Nous avons présenté dans ce chapitre l'architecture générale de *Context Aware Framework for Modeling and Simulation Mobile Cloud Application*, ainsi que la conception détaillée des outils qui le compose à savoir : *Mobile Cloud Architecture Description Language* (MC-ADL), Le middleware conscient au contexte *Smart Cloud Gate* (SCG) et le simulateur dédié pour l'étude et l'analyse du comportement des applications mobiles dans un environnement Cloud : *Mobile Cloud Simulation Toolkit* (MC-Sim). Nous avons présenté aussi, les détails techniques de la mise en œuvre du « *Contexte Aware Framework*

for Modeling and Simulation Mobile Cloud Application ». Nous avons essayé d'évaluer les outils du Framework et le comportement de l'application mobile à base de Cloud dans un environnement Cloud mobile changeant, en exécutants un scénario de simulation pour montrer le mécanisme de la reconfiguration de MC-Apps et l'évolution de son architecture dans deux états. Les résultats obtenus de la simulation ont renforcé nos arguments derrière notre proposition dans le domaine de mobile Cloud Computing.

CONCLUSION GÉNÉRALE & PERSPECTIVES

Mbile Cloud Computing est un paradigme qui fut apparaitre pour améliorer la performance des appareils mobiles de sorte qu'ils peuvent offrir des fonctionnalités similaires à celles offertes par les PCs. Les problématiques les plus abordées par le MCC sont : la prolongation de la durée de vie de la batterie, étendre la capacité de stockage et augmenter la puissance de traitement. Certes comme un paradigme récent, le MCC pose des défis restent à être affronté. Dans cette thèse, nous avons essayé de mettre en lumière ces problèmes, après une étude approfondie dans le domaine.

1- Dans le chapitre 1, nous avons étudié les méthodes et les technologies de développement des applications mobiles traditionnelles, afin de conclure que ces dernières sont développées d'une façon ad-hoc en suivant des tutoriels fournis par les fournisseurs des outils de développement. Également pour le développement des applications mobiles à base de Cloud (cf. Chapitre 2), ces dernières sont développées pour surmonter les limites des appareils mobiles et prendre avantage du Cloud, mais aussi d'une façon non méthodique et non plus en suivant un cycle de développement bien défini. Pour répondre à ces lacunes dans le domaine du MCC, nous avons proposé un Framework qui offre des outils et des moyens pour raisonner sur l'application mobile à base de Cloud avant son développement réel.

2- L'entrelacement avec le Cloud rend les applications mobiles plus compliquées. Afin d'entamer cette complexité, une méthode de développement doit être suivie. Dans le chapitre 3, nous avons donné une brève présentation des concepts de base de l'ingénierie dirigée par les modèles afin de démontrer les avantages apportés par la modélisation sur le développement des applications mobiles. Dans ce cadre, notre Framework proposé, fournis un outil de modélisation et de description des architectures des applications mobiles. Nous attestons qu'une application mobile performante est surement celle ayant une bonne conception (c.-à-d. représentation architecturale). Mobile Cloud Architecture Description Language (MC-ADL) est outil qui offre un langage de description pour modéliser et décrire les applications mobiles à base de Cloud (MC-Apps). L'architecture de MC-App étudié dans cette thèse est à base de composant où notre contribution principale dans ce niveau est de proposer des nouveaux types d'éléments -différents de ceux offerts par les ADLs traditionnels- qui permet de décrire la nature hétérogène de l'architecture de MC-App. MC-ADL est implémenté comme un plug-in sous éclipse offrant les fonctionnalités suivantes : une représentation graphique et textuelle de l'architecture de MC-App et un mécanisme d'évolutions architecturales en termes des transformations de modèles exécutées par le langage de transformation Atlas Transformation Langage (ATL).

3- L'objectif principal dans le domaine de Mobile Computing est de fournir des applications mobiles performantes et adaptables avec leurs environnements, dans notre cas (l'environnement Cloud et le contexte de l'appareil mobile). Dans le but d'exploiter les

informations contextuelles que nous jugeons très bénéfiques dans le développement des applications mobile, nous avons présenté un des outils de notre Framework, à savoir : un middleware sensible au contexte, Smart Cloud Gate (SCG). L'architecture proposée pour SCG favorise l'exploitation intelligente des informations contextuelles extraites du Cloud et du mobile au temps d'exécution et au niveau abstrait (niveau architectural).

Les applications conçues par notre Framework sont des applications réflectives qui prennent en charge leurs propres comportements en fonction du changement dans leurs contextes. La reflectiveness est abordée en proposant le concept de contrat « *MC-App's Contract »* entre l'application et son environnement, dont à son niveau, le comportement de l'application dans une situation donnée est défini. Le contrat est introduit au niveau architectural par le biais du modèle *MC-App Contract* décrit à l'aide d'un plug-in implémenté sous éclipse. Les informations contextuelles dans le contrat servent comme des paramètres d'évolution architecturale et de reconfiguration dynamique de l'application.

4- La nature du Cloud est extrêmement hétérogène et en plus payante, ce qui pose des difficultés pour tester et évaluer l'application mobile avant un déploiement réel. Alors, la solution est de tester et étudier le comportement de l'application mobile dans un environnement Cloud simulé. Malheureusement, l'étude établis dans le chapitre 4 qui porte sur les simulateurs Cloud à résulter que, les simulateurs existants n'offrent pas des moyens pour simuler le comportement d'une application mobile dans un environnement Cloud. En outre, un autre résultat de l'étude est que CloudSim est le simulateur le plus étendu et le plus simple à utiliser. Par conséquent, nous avons choisi d'ajouter des packages de simulation au CloudSim pour compléter ces manques et offrir le troisième outil de notre Framework: Mobile Cloud Simulation toolkit (MC-SIM). L'objectif de MC-Sim est d'offrir un environnement de simulation contrôlable et gratuit pour étudier répétitivement le comportement de l'application selon plusieurs scénarios. Il est à noter que, les entités de simulations sont générées à partir du modèle de l'architecture.

Le domaine du Mobile Cloud Computing est un domaine très vaste et encore récent, nous avons essayé dans cette thèse de répondre à quelques limites d'une perspective conceptuelle. Le travail de cette thèse aborde la modélisation et la simulation des applications mobiles à base de Cloud, mais nous avons aussi démontré l'importance des informations contextuelles en proposant seulement une architecture pour un middleware afin de démontrer comment gérer intelligemment ces informations. Certes, le travail proposé couvre une partie des problématiques dans l'exploitation des informations contextuelles, ce qui reste à faire comme futur travail sera présenté dans la section suivante.

Perspectives

Notre objective en réalisant ce travail de thèse est de proposer une démarche afin de rendre le développement des applications mobiles un processus efficace, méthodique et sophistiqué. En proposant notre Framework, nous avons concentré sur le côté

architectural. Par conséquent, il reste beaucoup de points à traiter pour compléter le Framework, à savoir :

1- Au niveau architectural nous nous sommes concentrés sur la structure interne de chaque élément ainsi que son type. Les relations entre ces éléments sont présentées d'une façon simple autant qu'un élément intermédiaire (c.-à-d. connecteur) caractérisé par les contraintes des éléments qui les relisent. Comme points d'extension dans ce niveau : une étude approfondie peut être établie sur la définition des connecteurs d'une façon détaillée sur, les propriétés non fonctionnelles de chaque élément et son interface, et enfin considérer la vérification du modèle architecturale.

2- Le niveau middleware regroupe plusieurs notions et concepts appartenant à plusieurs domaines. Nous avons proposé l'architecture pour le middleware Smart Cloud Gate. Alors, comme perspectives dans ce niveau reste la mise en œuvre de chaque composant de Smart Cloud Gate, notamment :

a) La découverte et la sélection des services Cloud

La première tâche exécutée par SCG est celle de son composant *Context analysis*. L'analyse du contexte comprend l'analyse de l'environnement mobile et du Cloud. Il existe plusieurs protocoles de découverte de service reste à choisir lequel à implémenter par *Context analysis* ou bien proposé une nouvelle approche comme un travail d'extension.

b) Algorithme de décision pour le déchargement de code

L'architecture que nous proposons pour les applications mobiles à base de Cloud favorise la composition hétérogène, où nous avons proposé trois types de composants dont leurs exécutions dépendent de plusieurs critères. Pour les composants exécutables au niveau de Cloud, leurs choix n'est pas fait aléatoirement, mais plutôt en exécutant une étude approfondis en termes d'algorithme de décision pour décider quand, quoi et comment décharger l'exécution du composant au Cloud. La perspective de cette section concerne l'implémentation d'un mécanisme de décision.

c) La composition

Nous avons signé précédemment qu'une application mobile ne peut se satisfaire par l'exploitation d'un seul service. Pour répondre à ce besoin, la composition des services hétérogènes est nécessaire. Plusieurs mécanismes de composition peuvent être proposés comme perspective de ce point.

d) L'adaptation

L'utilisation de plusieurs services Cloud fournis par de différentes plateformes et implémentant de différents API, demande une approche adaptative afin d'être utilisable par l'application. La perspective de ce point est la proposition d'une approche d'adaptation des services Cloud pour l'utilisation mobile.

e) L'orchestration des services des middlewares

La coordination des services Cloud utilisés, leurs organisations et leurs gestions sont faites par l'exécution d'un processus d'orchestration des services. Ce Dernier représente un autre point pas pris en considération dans cette thèse, et aussi une perspective pour de futurs travaux.

PUBLICATIONS PERSONNELLES

I- Conférences Internationales

Gherari, M., Amirat, A., & Oussalah, M. *Towards Smart Cloud Gate Middleware: An approach based on Profiling Technique.* In Conférence francophone sur les Architectures Logicielles (CAL).

Gherari, M., Amirat, A., Oussalah, M., & Laouar, R. *Towards a smart cloud gate for smart devices*. In Information Science and Technology (CIST), 2014 Third IEEE International Colloquium in (pp. 145-150). IEEE. Indexed Scopus & Dblp.

Gherari, M., Amirat, A., & Oussalah, M. *Describing Ubiquitous Mobile Cloud Systems*. IT4OD, 251. Tebessa 22-24 Octobre 2014.

II- Conférences Nationales

Gherari, M., Amirat, A., Oussalah, M., & Laouar, R., *Towards a Mobile Cloud Context-Aware Middleware*, Symposium on Complex Systems and Intelligent Computing (CompSIC), Souk-Ahras, 2015.

III-Journées Doctorales

Gherari, M., Amirat, A., Oussalah, M., & Laouar, R., *Description des architectures des applications mobiles Cloud*, Journées Ouvertes sur les Mathématiques et l'Informatique (JOMI 2014) Tébessa, Algérie.14-15 Mai 2014.

Gherari, M., Amirat, A., Oussalah, M., & Laouar, R. *Towards a Smart mobile Cloud applications development*, Journées Ouvertes sur les Mathématiques et l'Informatique (JOMI 2015) Tébessa, Algérie.25-26 Mai 2015.

IV-Revues International

Gherari, M., Amirat, A., Oussalah, M., & Laouar, R., *A smart mobile cloud environment for modelling and simulation of mobile cloud applications*; special issue on Mobile Cloud Computing, International journal of Embedded Systems (xxxx) N X, 00-000.

RÉFÉRENCES

G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith and P. Steggles (1999). *Towards a better understanding of context and context-awareness*. Handheld and ubiquitous computing, Springer.

P. ACCORD (2002). « *Etat de l'art sur les Langages de Description d'Architecture (ADLs).* » Disponible à l'adresse http://www. infres. enst. fr/projets/accord/lot1/lot_1: 1-2.

AEPONA (November 2010.). « *Mobile Cloud Computing Solution Brief.* » White Paper.

M. Ali (2009). *Green cloud on the horizon*. Cloud Computing, Springer**:** 451-459.

A. Amirat (2010). *Contribution à l'élaboration d'architectures logicielles à hiérarchies multiples*, Nantes.

A. Amirat and M. Oussalah (2009). « *First-class connectors to support systematic construction of hierarchical software architecture.* » Journal of Object Technology **8**(7): 107-130.

T. S. Andrew and M. van Steen (2007). "*Distributed systems-principles and paradigms (2.* »

Android (2016). « *https://www.android.com/.* »

P. Angin, B. Bhargava and S. Helal (2010). *A mobile-cloud collaborative traffic lights detector for blind navigation*. Mobile Data Management (MDM), 2010 Eleventh International Conference on, IEEE.

Apple (2008). « *https://www.apple.com/pr/library/2008/07/10iPhone-3G-on-Sale-Tomorrow.html.* »

I. Apple (2016). « *Available at: http://www.infinov.com/en/mobile/ios.* ».

Apps (2016). « *http://www.appsmiles.eu/2015/05/10-chiffres-sur-les-applications-mobiles/.*".

A. I. Avetisyan, R. Campbell, K. Lai, M. Lyons, D. S. Milojicic, H. Y. Lee, Y. C. Soh, N. K. Ming, J.-Y. Luke and H. Namgoong (2010). « *Open cirrus: A global cloud computing testbed.* » Computer(4): 35-43.

AwS3 (2016). « *Available at: http://aws.amazon.com/s3/* ».

M. Baldauf, S. Dustdar and F. Rosenberg (2007). « *A survey on context-aware systems.* » International Journal of Ad Hoc and Ubiquitous Computing **2**(4): 263-277.

O. Barais (2005). *Construire et Maîtriser l'évolution d'une architecture logicielle à base de composants*, Lille 1.

F. Barbier, C. Cauvet, M. Oussalah, D. Rieu, S. Bennasri and C. Souveyet (2002). « *Composants dans l'ingénierie des systèmes d'information: concepts clés et techniques de réutilisation.* » Actes des deuxièmes assises nationales du GDR-I3.

J. E. Bardram and T. R. Hansen (2010). « *Context-based workplace awareness.* » Computer Supported Cooperative Work (CSCW) **19**(2): 105-138.

L. Bass (2007). *Software architecture in practice*, Pearson Education India.

G. Belalem (2012). « *Towards optimisation of the management of resources in the CloudSim simulator.* » International Journal of Innovative Computing and Applications 1 **4**(1): 28-34.

C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan and D. Riboni (2010). « *A survey of context modelling and reasoning techniques.* » Pervasive and Mobile Computing **6**(2): 161-180.

X. Blanc (2005). « *MDA en action, Ingénierie logicielle guidée par les modèles, EYROLLES.* » Paris.

C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber and L. Tanca (2007). « *A data-oriented survey of context models.* » ACM Sigmod Record **36**(4): 19-26.

P. J. Brown, J. D. Bovey and X. Chen (1997). "*Context-aware applications: from the laboratory to the marketplace.* » Personal Communications, IEEE **4**(5): 58-64.

A. Buss (2002). *Simkit: component based simulation modeling with Simkit*. Proceedings of the 34th conference on Winter simulation: exploring new frontiers, Winter Simulation Conference.

R. Buyya and M. Murshed (2002). « *Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing.* » Concurrency and computation: practice and experience **14**(13-15): 1175-1220.

R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic (2009). "*Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility.* » Future Generation computer systems **25**(6): 599-616.

R. N. Calheiros, M. A. Netto, C. A. De Rose and R. Buyya (2013). « *EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications.* » Software: Practice and Experience **43**(5): 595-612.

R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose and R. Buyya (2011). « *CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms.* » Software: Practice and Experience **41**(1): 23-50.

R. Campbell, I. Gupta, M. Heath, S. Y. Ko, M. Kozuch, M. Kunze, T. Kwan, K. Lai, H. Y. Lee and M. Lyons (2009). *Open cirrusTM cloud computing testbed: federated data centers for open source systems and services research*. Proceedings of the 2009 conference on Hot topics in cloud computing, USENIX Association.

G. G. Castane, A. Nunez and J. Carretero (2012). *iCanCloud: A brief architecture overview*. Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on, IEEE.

CCM (2015) *Cycle de vie de logiciel, Available at: www.commentcamarche.net/contents/473-cycle-de-vie-d-un-logiciel*.

D. Chappell (2008). « *Introducing the Azure services platform.* » White paper, Oct **1364**(11).

G. Chen, B.-T. Kang, M. Kandemir, N. Vijaykrishnan, M. J. Irwin and R. Chandramouli (2004). « *Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices.* » Parallel and Distributed Systems, IEEE Transactions on **15**(9): 795-809.

G. Chen and D. Kotz (2000). *A survey of context-aware mobile computing research*, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

X. Chen, J. Liu, J. Han and H. Xu (2010). *Primary exploration of mobile learning mode under a cloud computing environment*. 2010 International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT).

R. Chow, M. Jakobsson, R. Masuoka, J. Molina, Y. Niu, E. Shi and Z. Song (2010). *Authentication in the clouds: a framework and its application to mobile users*. Proceedings of the 2010 ACM workshop on Cloud computing security workshop, ACM.

B. Combemale (2008). « *Ingénierie Dirigée par les Modèles (IDM) État de l'art.* » Management: 1-19.

J. R. Cooperstock, K. Tanikoshi, G. Beirne, T. Narine and W. A. Buxton (1995). *Evolution of a reactive environment*. Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press/Addison-Wesley Publishing Co.

E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra and P. Bahl (2010). *MAUI: making smartphones last longer with code offload*. Proceedings of the 8th international conference on Mobile systems, applications, and services, ACM.

K. Czarnecki and S. Helsen (2003). *Classification of model transformation approaches*. Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, USA.

Y.-S. Dai, M. Xie and X. Wang (2007). « *A heuristic algorithm for reliability modeling and analysis of grid systems.* » Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on **37**(2): 189-200.

N. Davies, K. Mitchell, K. Cheverst and G. Blair (1998). *Developing a context sensitive tourist guide*. 1st Workshop on Human Computer Interaction with Mobile Devices, GIST Technical Report G98-1.

J. W. Davis (1993). *Power benchmark strategy for systems employing power management*. Electronics and the Environment, 1993., Proceedings of the 1993 IEEE International Symposium on, IEEE.

B. Desnos (Decembre 2011). « *Livre Blanc Comparatif des solutions de développement multiplateforme mobile* ».

A. K. Dey (2001). « *Understanding and using context.* » Personal and ubiquitous computing **5**(1): 4-7.

S. Diaw, R. Lbath and B. Coulette (2010). « *État de l'art sur le développement logiciel basé sur les transformations de modèles.* » Technique et Science Informatiques **29**(4-5): 505-536.

H. T. Dinh, C. Lee, D. Niyato and P. Wang (2011). "*A survey of mobile cloud computing: architecture, applications, and approaches.* » Wireless Communications and Mobile Computing.

H. T. Dinh, C. Lee, D. Niyato and P. Wang (2013). "*A survey of mobile cloud computing: architecture, applications, and approaches.* » Wireless Communications and Mobile Computing **13**(18): 1587-1611.

C. Doukas, T. Pliakas and I. Maglogiannis (2010). *Mobile healthcare information management utilizing Cloud Computing and Android OS*. Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE, IEEE.

C. L. Dumitrescu and I. Foster (2005). *GangSim: a simulator for grid scheduling studies*. Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on, IEEE.

F. Dupont (2013) *5 bonnes raisons du MDA pour le développement Mobile,*.

EC2 (2010). *Amazon Elastic Compute Cloud (EC2), Available at: http://www.amazon.com/ec2/ [Last Access 15-4-2016].*

Éclipse (2016). « *Available at: https://eclipse.org/.* »

Ekito (2012). « *Ekito, Application mobile : web ou natif. 2012.*".

S. Elrod, G. Hall, R. Costanza, M. Dixon and J. Des Rivieres (1993). « *Responsive office environments.* » Communications of the ACM **36**(7): 84-85.

eugenia (2016) *available at : http://www.eclipse.org/epsilon/doc/eugenia/*.

R. Ferzli and I. Khalife (2011). *Mobile cloud computing educational tool for image/video processing algorithms*. Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE), 2011 IEEE, IEEE.

Flickr (2016). « *http://www.flickr.com/* ».

H. Flores and S. N. Srirama (2013). « *Mobile cloud middleware.* » Journal of Systems and Software.

G. H. Forman and J. Zahorjan (1994). « *The challenges of mobile computing.* » Computer **27**(4): 38-47.

M. Forum (2016). « *http://www.mobilecloudcomputingforum.com/* ».

GAE (2010) *Google App Engine, Available at: http://appengine.google.com [Last Access 5-4-2016]*.

H.-q. Gao and Y.-j. Zhai (2010). *System design of cloud computing based on mobile learning*. Knowledge Acquisition and Modeling (KAM), 2010 3rd International Symposium on, IEEE.

A. Garcia and H. Kalva (2011). *Cloud transcoding for mobile video content delivery*. Proceedings of the IEEE International Conference on Consumer Electronics (ICCE).

S. K. Garg and R. Buyya (2011). *Networkcloudsim: Modelling parallel applications in cloud simulations*. Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on, IEEE.

R. Grossman, Y. Gu, M. Sabala, C. Bennet, J. Seidman and J. Mambratti (2009). "*The open cloud testbed: A wide area testbed for cloud computing utilizing high performance network services.* » arXiv preprint arXiv:0907.4810.

T. Gu, H. K. Pung and D. Q. Zhang (2004). *A middleware for building context-aware mobile services*. Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th, IEEE.

J. Gustedt, E. Jeannot and M. Quinson (2009). "*Experimental methodologies for large-scale systems: a survey.* » Parallel Processing Letters **19**(03): 399-418.

D. B. Hoang and L. Chen (2010). *Mobile cloud for assistive healthcare (MoCAsH)*. Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific, IEEE.

C. Hoareau and I. Satoh (2009). "*Modeling and processing information for context-aware computing: A survey.* » New Generation Computing **27**(3): 177-196.

F. Howell and R. McNab (1998). « *SimJava: A discrete event simulation library for java.* » Simulation Series **30**: 51-56.

D. Huang (2011). « *Mobile cloud computing.* » IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter **6**(10): 27-31.

G. Huerta-Canepa and D. Lee (2010). *A virtual cloud computing provider for mobile devices*. Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, ACM.

W. Itani, A. Kayssi and A. Chehab (2010). *Energy-efficient incremental integrity for securing storage in mobile cloud computing*. Energy Aware Computing (ICEAC), 2010 International Conference on, IEEE.

iwebyou (2015). « *http://www.iwebyou.fr/actualites/statistiques-internet-janvier-2015/.* »

M.-H. J. Boccuzzi and M. Ruggiero (2011) *"Femtocells: design & application, »,*. .

M. Jakobsson, E. Shi, P. Golle and R. Chow (2009). *Implicit authentication for mobile devices*. Proceedings of the 4th USENIX conference on Hot topics in security, USENIX Association.

X. Jin and Y.-K. Kwok (2010). *Cloud assisted P2P media streaming for bandwidth constrained mobile subscribers*. Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on, IEEE.

F. Jouault, F. Allilaire, J. Bézivin, I. Kurtev and P. Valduriez (2006). *ATL: a QVT-like transformation language*. Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications, ACM.

E. Jung, Y. Wang, I. Prilepov, F. Maker, X. Liu and V. Akella (2010). *User-profile-driven collaborative bandwidth sharing on mobile phones*. Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, ACM.

R. Kakerow (2002). *Low power design methodologies for mobile communication*. Computer Design: VLSI in Computers and Processors, 2002. Proceedings. 2002 IEEE International Conference on, IEEE.

I. Kelényi and J. K. Nurminen (2010). *Cloudtorrent-energy-efficient bittorrent content sharing for mobile devices via cloud services*. Proceedings of the 7th IEEE on Consumer Communications and Networking Conference (CCNC).

A. Klein, C. Mannweiler, J. Schneider and H. D. Schotten (2010). *Access schemes for mobile cloud computing*. Mobile Data Management (MDM), 2010 Eleventh International Conference on, IEEE.

D. Kliazovich, P. Bouvry and S. U. Khan (2012). « *GreenCloud: a packet-level simulator of energy-aware cloud computing data centers.* » The Journal of Supercomputing **62**(3): 1263-1283.

D. Kliazovich, P. Bouvry and S. U. Khan (2013). *Simulation and performance analysis of data intensive and workload intensive cloud computing data centers*. Optical Interconnects for Future Data Center Networks, Springer**:** 47-63.

L. T. Kohn, J. M. Corrigan and M. S. Donaldson (2000). *To err is human:: building a Safer Health System*, National Academies Press.

F. Kon, F. Costa, G. Blair and R. H. Campbell (2002). « *The case for reflective middleware.* » Communications of the ACM **45**(6): 33-38.

D. Kopec, M. Kabir, D. Reinharth, O. Rothschild and J. Castiglione (2003). "*Human errors in medical practice: systematic classification and reduction with automated information systems.* » Journal of medical systems **27**(4): 297-313.

U. Kremer, J. Hicks and J. Rehg (2001). *A compilation framework for power and energy management on mobile computers*. Languages and Compilers for Parallel Computing, Springer**:** 115-131.

K. Kumar and Y.-H. Lu (2010). "*Cloud computing for mobile users: Can offloading computation save energy?* » Computer(4): 51-56.

Laine (2013) *Généralités sur l'approche MDA*.

A. T. Language (2016). « *avaibale at: https://eclipse.org/atl/.* »

A. Legrand, L. Marchal and H. Casanova (2003). *Scheduling distributed applications: the simgrid simulation framework*. Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on, IEEE.

Z. Leina, P. Tiejun and Y. Guoqing (2010). *Research of mobile security solution for fourth party logistics*. Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on, IEEE.

H. Li and X.-S. Hua (2010). *Melog: mobile experience sharing through automatic multimedia blogging*. Proceedings of the 2010 ACM multimedia workshop on Mobile cloud media computing, ACM.

J. Li (2010). *Study on the development of mobile learning promoted by cloud computing*. Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference on, IEEE.

Y.-C. Li, I.-J. Liao, H.-P. Cheng and W.-T. Lee (2010). *A cloud computing framework of free view point real-time monitor system working on mobile devices*. Intelligent Signal Processing and Communication Systems (ISPACS), 2010 International Symposium on, IEEE.

Z. Li, C. Wang and R. Xu (2001). *Computation offloading to save energy on handheld devices: a partition scheme*. Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems, ACM.

S.-H. Lim, B. Sharma, G. Nam, E. K. Kim and C. R. Das (2009). *MDCSim: A multi-tier data center simulation, platform*. Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on, IEEE.

L. Liu, R. Moulic and D. Shea (2010). *Cloud service portal for mobile device management*. e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on, IEEE.

C. Mannweiler, A. Klein, J. Schneider and H. D. Schotten (2009). *Exploiting user and network context for intelligent radio network access*. Ultra Modern Telecommunications & Workshops, 2009. ICUMT'09. International Conference on, IEEE.

R. N. Mayo and P. Ranganathan (2003). *Energy consumption in mobile devices: why future systems need requirements–aware energy scale-down*. Power-Aware Computer Systems, Springer**:** 26-40.

P. K. McKinley, F. A. Samimi, J. K. Shapiro and C. Tang (2006). *Service clouds: a distributed infrastructure for constructing autonomic communication services*. Dependable, Autonomic and Secure Computing, 2nd IEEE International Symposium on, IEEE.

N. Medvidovic and R. N. Taylor (2000). « *A classification and comparison framework for software architecture description languages.* » Software Engineering, IEEE Transactions on **26**(1): 70-93.

P. Mell and T. Grance (2011). « *The NIST Definition of Cloud Computing–NIST Special Publication, Reports on Computer Systems Technology.* » National Institute of Standards and Technology, Gaithersburg, MD, USA.

S. J. Mellor, T. Clark and T. Futagami (2003). « *Model-driven development: guest editors » introduction.* » IEEE software **20**(5): 14-18.

R. Minelli and M. Lanza (2013). *Software analytics for mobile applications--insights & lessons learned*. Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on, IEEE.

M. Mobile (2016). « *http://www.idc.com/prodserv/smartphone-market-share.jsp.* »

R. Nick, J. Pascoe and D. Morse (1997). "*Enhanced reality fieldwork: the contextaware archaeologist assistant.* » Computer Applications & Quantitative Methods in Archaeology, volume 0. Archaeopress.

M. Nkosi and F. Mekuria (2010). *Cloud computing for enhanced mobile health applications*. Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, IEEE.

A. Nunez, G. Castañé, J. Vázquez-Poletti, A. Caminero, J. Carretero and I. Llorente (2011). *Design of a flexible and scalable hypervisor module for simulating cloud computing environments*. Performance Evaluation of Computer & Telecommunication Systems (SPECTS), 2011 International Symposium on, IEEE.

A. Nuñez, J. L. Vázquez-Poletti, A. C. Caminero, J. Carretero and I. M. Llorente (2011). *Design of a new cloud computing simulation platform*. Computational Science and Its Applications-ICCSA 2011, Springer**:** 582-593.

A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero and I. M. Llorente (2012). « *iCanCloud: A flexible and scalable cloud infrastructure simulator.* » Journal of Grid Computing **10**(1): 185-209.

J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn and F. Jahanian (2008). *Virtualized in-cloud security services for mobile devices*. Proceedings of the First Workshop on Virtualization in Mobile Computing, ACM.

onehourtranslation (2016). « *http://www.onehourtranslation.com/.* »

S. Ou, K. Yang, A. Liotta and L. Hu (2007). *Performance analysis of offloading systems in mobile wireless environments*. Communications, 2007. ICC'07. IEEE International Conference on, IEEE.

P. Papakos, L. Capra and D. S. Rosenblum (2010). *Volare: context-aware adaptive cloud service discovery for mobile systems*. Proceedings of the 9th International Workshop on Adaptive and Reflective Middleware, ACM.

J. Pascoe (1998). *Adding generic contextual capabilities to wearable computers*. Wearable Computers, 1998. Digest of Papers. Second International Symposium on, IEEE.

L. D. Paulson (2003). « *Low-power chips for high-powered handhelds.* » Computer **36**(1): 21-23.

PC-world (2010). « *http://www.pcworld.com/article/161410/article.html.* »

D. Popovici (2012). *Gestion du contexte pour des applications mobiles dédiées aux transports*, Université de Valenciennes et du Hainaut-Cambresis.

A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam and N. Sharma (2009). *Towards autonomic workload provisioning for enterprise grids and clouds*. Grid Computing, 2009 10th IEEE/ACM International Conference on, IEEE.

rabidvaluesolution (2016). « *www.rapidvaluesolutions.com.* »

R. Rieger and G. Gay (1997). *Using mobile computing to enhance field study*. Proceedings of the 2nd international conference on Computer support for collaborative learning, International Society of the Learning Sciences.

M. Rosenblum and J. K. Ousterhout (1992). « *The design and implementation of a log-structured file system.* » ACM Transactions on Computer Systems (TOCS) **10**(1): 26-52.

A. Rudenko, P. Reiher, G. J. Popek and G. H. Kuenning (1998). « *Saving portable computer battery power through remote process execution.* » ACM SIGMOBILE Mobile Computing and Communications Review **2**(1): 19-26.

J. Rumbaugh, I. Jacobson and G. Booch (2004). *Unified Modeling Language Reference Manual, The*, Pearson Higher Education.

N. Sadou-Harireche (2007). *Evolution Structurelle dans les Architecture Logicielles à base de Composants*, PhD thesis, Université de Nantes.

F. A. Samimi, P. K. McKinley and S. M. Sadjadi (2006). *Mobile service clouds: a self-managing infrastructure for autonomic mobile computing services*. Self-Managed Networks, Systems, and Services, Springer**:** 130-141.

F. A. Samimi, P. K. McKinley, S. M. Sadjadi and P. Ge (2004). *Kernel-middleware interaction to support adaptation in pervasive computing environments*. Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing, ACM.

M. Satyanarayanan (1996). *Fundamental challenges in mobile computing*. Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing, ACM.

M. Satyanarayanan (2010). *Mobile computing: the next decade,|| in Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (MCS)*, June.

B. N. Schilit and M. M. Theimer (1994). « *Disseminating active map information to mobile hosts.* » Network, IEEE **8**(5): 22-32.

J. Shen, S. Yan and X.-S. Hua (2010). *The e-recall environment for cloud based mobile rich media data management*. Proceedings of the 2010 ACM multimedia workshop on Mobile cloud media computing, ACM.

Shozu (2016). « *http://www.shozu.com/portal/index.do* ».

Sim (2016) *Simulation Informatique, Available at: http://www.futura-sciences.com/magazines/high-tech/infos/dico/d/informatique-simulation-informatique-11319/*.

A. Smailagic and M. Ettus (2002). *System design and power optimization for mobile computers*. isvlsi, IEEE.

Z. Song, J. Molina, S. Lee, H. Lee, S. Kotani and R. Masuoka (2009). *Trustcube: An infrastructure that builds trust in client*. Future of Trust in Computing, Springer**:** 68-79.

I. Sriram (2009). *SPECI, a simulation tool exploring cloud-scale data centres*. Cloud Computing, Springer**:** 381-392.

Statista (2016). « *http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.* »

Stéphane (2015) *Nombreux sont les développeurs qui n'ont jamais eu à concevoir une application mobile*

D'après les résultats d'une enquête.

L. Sweeney (2002). « *k-anonymity: A model for protecting privacy.* » International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **10**(05): 557-570.

M. Tang and J. Cao (2006). *A dynamic mechanism for handling mobile computing environmental changes*. Proceedings of the 1st international conference on Scalable information systems, ACM.

W.-T. Tang, C.-M. Hu and C.-Y. Hsu (2010). *A mobile phone based homecare management system on the cloud*. Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on, IEEE.

W.-T. Tsai, X. Sun and J. Balasooriya (2010). *Service-oriented cloud computing architecture*. Information Technology: New Generations (ITNG), 2010 Seventh International Conference on, IEEE.

U. Varshney (2007). « *Pervasive healthcare and wireless health monitoring.* » Mobile Networks and Applications **12**(2-3): 113-127.

E. Vartiainen and K. Väänänen-Vainio-Mattila (2010). *User experience of mobile photo sharing in the cloud*. Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia, ACM.

C. Vecchiola, X. Chu and R. Buyya (2009). « *Aneka: a software platform for .NET-based cloud computing.* » High Speed and Large Scale Scientific Computing **18**: 267-295.

M. Walsh (2010). « *Gartner: Mobile to outpace desktop web by 2013.* » Online Media Daily.

C. Wang and Z. Li (2004). « *A computation offloading scheme on handheld devices.* » Journal of Parallel and Distributed Computing **64**(6): 740-746.

Q. Wang, L. Ren and L. Zhang (2011). *Design and implementation of virtualization-based middleware for cloud simulation platform*. C, The 4th IEEE International Conference on Computer Science and Information Technology.

R. Want, A. Hopper, V. Falcao and J. Gibbons (1992). « *The active badge location system.* » ACM Transactions on Information Systems (TOIS) **10**(1): 91-102.

J. B. Warmer and A. G. Kleppe (1998). "*The Object Constraint Language: Precise Modeling With Uml (Addison-Wesley Object Technology Series).* »

B. Wickremasinghe (2009). « *CloudAnalyst: A CloudSim-based tool for modelling and analysis of large scale cloud computing environments.* » MEDC project report **22**(6): 433-659.

B. Wickremasinghe, R. N. Calheiros and R. Buyya (2010). *Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications*. Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, IEEE.

X. Yang, T. Pan and J. Shen (2010). *On 3G mobile e-commerce platform based on cloud computing*. Ubi-media Computing (U-Media), 2010 3rd IEEE International Conference on, IEEE.

Z. Ye, X. Chen and Z. Li (2010). *Video based mobile location search with large set of SIFT points in cloud*. Proceedings of the 2010 ACM multimedia workshop on Mobile cloud media computing, ACM.

C. Yin, B. David and R. Chalon (2009). *Use your mobile computing devices to learn-Contextual mobile learning system design and case studies*. Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on, IEEE.

L. Zhang, X. Ding, Z. Wan, M. Gu and X.-Y. Li (2010). *WiFace: a secure geosocial networking system using WiFi-based multi-hop MANET*. Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, ACM.

H. Zhangwei and X. Mingjun (2010). *A distributed spatial cloaking protocol for location privacy*. Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on, IEEE.

W. Zhao, Y. Sun and L. Dai (2010). *Improving computer basis teaching through mobile communication and cloud computing technology*. Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, IEEE.

W. Zhenyu, Z. Chunhong, J. Yang and W. Hao (2010). *Towards cloud and terminal collaborative mobile social network service*. Social Computing (SocialCom), 2010 IEEE Second International Conference on, IEEE.

P. Zou, C. Wang, Z. Liu and D. Bao (2010). *Phosphor: A cloud based DRM scheme with sim card*. Web Conference (APWEB), 2010 12th International Asia-Pacific, IEEE.