UNIVERSITE SULTAN MOULAY SLIMANE

Faculté des Sciences et Techniques

Béni-Mellal

Formation Doctorale : **Mathématiques et Physique Appliquée**

THÈSE
Présentée par

FATIHA AIT YACINE

Pour l'obtention du grade de
DOCTEUR

Spécialité **: INFORMATIQUE**

Option **: INFORMATIQUE**

Modélisation et Implémentation des Systèmes Multi-Agents : Application à l'Environnement Numérique de Travail.

Soutenue le Lundi 17 Juin 2019 à 09h devant la commission d'examen

Said SAFI	Professeur à la Faculté Polydisciplinaire, Béni Mellal	Président
Mohamed BASLAM	Professeur à la Faculté des Sciences et Techniques, Béni Mellal	Rapporteur
Mohamed FAKIR	Professeur à la Faculté des Sciences et Techniques, Béni Mellal	Rapporteur
Benayad NSIRI	Professeur à L'Ecole Normale Supérieure de L'Enseignement Technique, Rabat	Rapporteur
Rachid EL AYACHI	Professeur à la Faculté des Sciences et Techniques, Béni Mellal	Examinateur
Noureddine ASKOUR	Professeur à la Faculté des Sciences et Techniques, Béni Mellal	Directeur de thèse
Belaid BOUIKHALENE	Professeur à la Faculté Polydisciplinaire, Béni Mellal,	Co-directeur de thèse

Résumé

Les progrès considérables dans les domaines des technologies de l'information et de la communication plus la distribution et l'hétérogénéité des systèmes d'information, ont donné naissance à l'intelligence Artificielle (IAD) et les systèmes multi-agent (SMA) qui sont actuellement très largement utilisés particulièrement pour les applications distribuées complexes nécessitant l'interaction entre plusieurs entités hétérogènes. Les solutions proposées par les systèmes multi-agent sont prometteuses et permettent d'obtenir des systèmes flexibles et évolutifs.

Ce travail de recherche porte sur le développement des applications distribuées, extensibles et adaptables aux systèmes d'information, de leurs intégrations en faisant intervenir les systèmes multi-agent à base du paradigme agent et les systèmes distribués à base du paradigme objet, qui présentent deux technologies ayant un grand impact sur le développement des applications distribuées. L'extensibilité et l'adaptabilité s'imposent de plus en plus comme caractéristiques importantes et essentielles dans plusieurs systèmes informatiques vus la popularisation de l'internet et des technologies web. Dans ce contexte et afin d'automatiser et globaliser des travaux administratifs universitaires que contribue la thématique de cette thèse. Pour utiliser ces deux disciplines, nous avons tout d'abord abouti quelques études théoriques de l'évolution des paradigmes de développement des applications distribuées, en particulier l'agent, qui représente le pilier des SMA, et la distribution à base du paradigme objet. Nous avons présenté leurs cadres méthodologiques et techniques, en suite nous avons réalisé une étude comparative sur le niveau d'abstraction et de dynamisme et flexibilité de ces paradigmes pour pouvoir faire un bon choix. Nous avons pris en compte ces aspects théoriques pour automatiser des travaux administratifs du service de scolarité universitaire en réalisant dans un premier temps une application distribuée à base du paradigme objet et l'architecture de distribution 3-tiers, et ensuite deux modèles d'application à base des SMA.

Mots Clés : Intelligence Artificielle (IA), Intelligence Artificielle Distribuée (IAD), System Multi-Agents(SMA), Agent, ENT, Apogee, USMS, J2EE, JSP, Servlet, API, MVC, UML, JADE ; Spring Boot ; DAO ; Hibernate.

Abstract

The considerable progress in the areas of information technology, communication, the distribution and the heterogeneity of information systems, have led to birth of the Artificial Intelligence (IAD) and the systems multi-agent (SMA) who are currently very widely used, specially for complexes distributed Applications requiring the interaction between several heterogeneous entities. The solutions proposed by the systems multi-agent are promising and allow you to get the systems flexible and scalable.

This research work focuses on the development of distributed applications, which are extensible and adaptable to the information system of their integrations by involving the multi-agent systems SMA at base of the agent paradigm and distributed systems based on the object paradigm, which present two technologies having a great impact on the development of distributed applications. The scalability and adaptability are needed more and more in several computer systems saw the popularization of the internet and web technologies. In this context and for the automation and globalization administrative work of university education, contributes the theme of this thesis. To use these two disciplines, we have at first succeeded theoretical studies on the evolution of the distributed application development paradigm, in particular the agent, which is the mainstay of the SMA and the distribution on basis of the object paradigm. We presented their methodological and technical frameworks, and then we realized a comparative study on the level of abstraction and dynamism of these paradigms. We have taken into account these theoretical aspects to automate the administrative work of the service of the university education by performing in the first time a distributed application on the basis of the object paradigm and the distribution architecture 3-thirds, then two models of application on the basis of multi-agent system.

Key words: Artificial Intelligence (IA), Distributed Artificial Intelligence (IAD), System multi-agents (SMA), agent. ENT, Apogee, USMS, J2EE, JSP, Servlet, API, MVC, UML, JADE; Spring Boot; DAO; Hibernate.

Remerciements

Je tiens à adresser mes remerciements et à exprimer ma profonde admiration et gratitude aux personnes qui m'ont apporté leur aide et qui ont ainsi contribué à l'élaboration, et favorisé l'aboutissement de ce projet.

J'adresse mes plus vifs remerciements à Mr. Pr Noureddine ASKOUR, mon encadrant, qui a fait preuve de disponibilité à chaque fois que j'avais besoin de son soutien. Son encadrement et ses conseils ont été d'un appui considérable.

Toute ma profonde gratitude à Monsieur Pr. Belaid BOUIKHALENE, mon Co encadrant dans ce projet, pour l'aide et le temps qu'il m'a consacré à ma disposition durant la période de ce travail.
Je tiens à remercier aussi les membres du jury, pour avoir accepté de juger notre modeste travail.

Enfin, je remercie de tout mon cœur toute ma famille et amis pour leur soutien et leur aide inconditionnels ainsi que leur présence inestimable à mes côtés durant toute la durée de cette thèse. Merci à vous tous.

Table des matières

Liste des figures

Liste de tables

Glossaire des Acronymes

API	Application Programming Interface
J2EE	Java Enterprise Edition
JSP	Java Server Pages
MVC	Model-View-Controller
TIC	Technologies de l'Information et de la Communication
JVM	Java Virtual Machine
UML	Unified Modeling Language
SGBD	Système de Gestion de Base de Données
ACL	Agent Communications Language
IA	Intelligence Artificielle
IAD	Intelligence Artificielle Distribuée
SMA	Système Multi Agent
APOGEE	Application pour l'organisation et la gestion des enseignements et les étudiants
AMUE	Agence de mutualisation des universités et des établissements
TIC	Technologies de l'Information et de la Communication

JVM	Java Virtual Machine
AGR	Agent, Group, Rôle
M-UML	Mobile Unified Modeling Language
A-UML	Agent Unified Modeling Language
JPA	Java Persistance API
DAO	Data Access Objet

Introduction Générale

Face à la complexité croissante des applications informatiques, due à la croissance de leurs tailles et à l'hétérogénéité des systèmes d'information, ainsi que pour construire des applications plus ouvertes et dynamiques, les chercheurs ont débuté des recherches depuis plus d'une vingtaine d'années pour trouver de nouveaux concepts de développement pour enrichir l'ingénierie logicielle et lui permettre de faire son rôle académique et industriel. Ceci consiste à fournir les modèles et les techniques adéquats qui permettent de gérer efficacement cette complexité et de faciliter la construction des applications de haute qualité caractérisées par un nombre important de composants avec des schémas d'interactions et des comportements généraux dynamiques avec le changement de l'environnement, des services et des utilisateurs.

Plusieurs paradigmes du génie logiciel, ont tenté d'apporter des solutions à ce problème, mais semblent avoir échoué à deux niveaux essentiels : d'une part, ils manquent de flexibilité dans la manière de définir les interactions entre les différentes entités logicielles, et d'une autre part ils ne fournissent pas suffisamment de moyens pour représenter les structures organisationnelles [1]. Dans cette optique, La recherche a donc conduit à développer des systèmes composés de plusieurs entités parmi lesquelles sont réparties des tâches complexes à exécuter. Cette approche de conception de systèmes, appelés approche distribuée, s'est développée dans les différentes branches de l'informatique. Ainsi, à partir de l'Intelligence Artificielle (IA) est née l'Intelligence Artificielle Distribuée (lAD) qui a elle-même engendré les Systèmes Multi-Agents (SMA).

L'intelligence Artificielle (IA) est reconnue comme étant une discipline informatique qui a pour objectif de modéliser ou de simuler des comportements humains dits intelligents tels que la perception, la prise de décision, la compréhension, l'apprentissage, etc. Elle s'attache à la construction des programmes informatiques, capables d'exécuter des tâches complexes, en s'appuyant sur une centralisation et une concentration de l'intelligence au sein d'un système unique. Cependant l'IA a vite rencontré un certain nombre de difficultés, dues pour la plupart à la nécessité d'intégrer, au sein d'une même base de connaissances, l'expertise, les compétences et les connaissances d'individus différentes qui, dans la réalité, communiquent et collaborent pour la réalisation d'un but commun.

L'Intelligence Artificielle Distribuée (IAD)[2] est née, au début des années 80, de la volonté de remédier aux insuffisances et d'enrichir l'approche classique de l'IA en proposant la distribution de l'expertise sur un groupe d'agent, non soumis à un contrôle centralisé qui devront être capables de travailler et d'agir dans un environnement commun et de résoudre les conflits éventuels. En résumé, l'IAD s'intéresse entre autres à la modélisation des comportements intelligents qui sont le produit de l'activité coopérative entre plusieurs agents, d'où la réalisation des systèmes dit « **multi-agent** ».

Le principe d'un modèle IAD réside dans le fait de découper le problème à résoudre en une multitude de sous-problèmes qui vont être à leur tour résolus par des agents afin de fournir des solutions partielles. Ces solutions partielles vont ensuite être regroupées pour construire la solution globale du problème.

La figure 1 présente les trois axes fondamentaux dans la recherche en I.A.D :

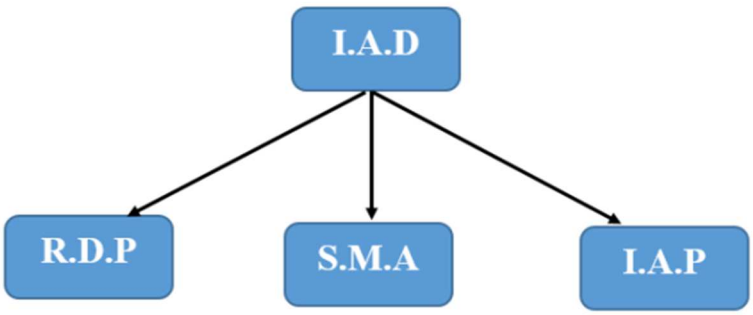


Figure 1: Axes de l'intelligence Artificielle Distribuée

- La résolution distribuée des problèmes (R.D.P) : elle s'intéresse à la manière de diviser un problème particulier sur un ensemble d'entités distribuées. Elle s'intéresse aussi `à la manière de partager la connaissance du problème et d'en obtenir la solution [3].

- Les systèmes multi-agents (S.M.A) : ils s'agissent de faire coopérer un ensemble d'agents dotés d'un comportement intelligent et de coordonner leurs buts et leurs plans d'actions pour la résolution d'un problème. C'est le thème auquel nous nous intéressons le plus dans ce travail de recherche [4].

- L'intelligence artificielle parallèle (I.A.P) : elle concerne le développement de langages et d'algorithmes parallèles pour l'I.A.D. L'I.A.P

visent l'amélioration des performances des systèmes d'intelligence artificielle sans, toutefois, s'intéresser à la nature du raisonnement ou au comportement intelligent d'un groupe d'agents. Cependant, il est vrai que le développement de langages concurrents et d'architectures parallèles peut avoir un impact important sur les systèmes d'I.A. D [5].

Contexte et problématique

A cause de l'hétérogénéité et la distribution des systèmes d'information, ainsi que l'évolution de la technologie internet et des réseaux d'entreprises, qui a contribué à automatiser et globaliser les processus administratifs dans les entreprises et les établissements publiques, on a tendance aujourd'hui à dépasser le monde des logiciels statiques et monopostes (basés dans une seule machine isolée) vers les logiciels distribués et dynamiques. La particularité à prendre en compte lors du développement de telles applications c'est que les composantes de ces dernières sont à exécuter sur des machines distantes les unes des autres et qui communiquent via des réseaux. Cette particularité fait des applications distribuées des applications plus complexes que les applications centralisées.

L'intelligence artificielle distribuée, à base des SMA (paradigme agent), et les systèmes distribués (paradigme objet) sont deux disciplines de recherches très importantes et qui sont toutes les deux bien équipées de moyens et d'outils leurs permettant de contribuer considérablement dans le développement d'applications distribuées, ouvertes et adaptables.

Dans le contexte décrit ci-dessus, nous nous sommes intéressés dans nos travaux au développement d'applications distribuées en utilisant les deux disciplines, les SMA et les systèmes distribuées, afin de réaliser des modèles d'applications dynamiques et adaptables au système d'information universitaire qui est distribué et évolutif. Ces modèles automatisent des travaux administratifs de l'université pour dépasser le mode des applications statiques monopostes et les processus administratifs classiques.

Structure du document

La suite de ce document est organisée en cinq chapitres de la façon suivante :

Chapitre 1 constitue un état de l'art des agents et systèmes multi-agents auxquels nous ferons largement appel tout au long de cette thèse et qui présentent l'une des axes de recherche en I.A.D. Nous décrivons les agents et les systèmes-multi agents, ainsi que leurs propriétés, liées à leurs environnements, organisations et langages de communication, ces propriétés leurs donnent une importance dans la résolution des problèmes complexes, en particulier le développement des applications distribuées.

Chapitre 2 décrit brièvement l'évolution de quelques paradigmes de développement des logiciels, qui représente des piliers de développements des applications dynamiques et distribuées, ainsi que les méthodologies qui leurs sont appropriées et les outils qui sont évolués pour faciliter l'utilisation de ces paradigmes. Nous nous focalisons dans notre travail sur les méthodologies orientées objet et les méthodologies orientées agent. Ce chapitre présente aussi une analyse comparative entre les paradigmes décrits, cette comparaison nous a permis de voir les grands avantages des SMA dans la réalisation des applications distribuées, flexibles et dynamiques.

Chapitre 3 la première partie de ce chapitre est consacrée à une description détaillée de la problématique et des objectifs de cette thèse. Par la suite, nous présentons notre première contribution pour la résolution de cette problématique, il s'agit d'une application « Scolarité Online » à base d'une architecture distribuées 3-tiers et du paradigme objet. L'insuffisance de dynamisme et flexibilité de cette contribution, nous a poussés vers une deuxième contribution à base des SMA.

Chapitre 4 décrit un modèle multi-agent simple qui permet l'automatisation de quelques taches administratives de la scolarité.

Chapitre 5 présente un deuxième modèle SMA « Scolarité intelligente » qui est plus dynamique, flexible et adaptable à l'architecture du système d'information de notre université.

Ces 2 modèles multi-agent se présentent sous formes d'applications autonomes qui intègrent des entités agents coopératifs et communicatifs entre eux pour réaliser leurs objectifs. Elles représentent un cas d'étude des SMA dans la résolution des problèmes administratifs complexes et distribués.

En conclusion, nous terminons ce mémoire de thèse par une conclusion générale, qui récapitule les travaux réalisés et fait le point sur un ensemble de perspectives envisagées.

Les Travaux publiés

Les publications :

- JADE Multi-Agent Middleware Applied to Contribute to Certificate Management of Students

TELKOMNIKA Indonesian Journal of Electrical Engineering Vol. 16, No. 1, October 2015, pp. 176 ~ 181 DOI: 10.11591/telkomnika.v16i1.8672.

Fatiha Ait yacine*, Badr Hssina, Belaid Bouikhalene, Sultan MoulaySlimane University, Morocco.

Les communications

- Virtual Learning Environment Mobile and Multi Agents. The University Sultan Moulay Slimane Case Study

The 5th International Conference on Business Intelligence, April 17 - 19, 2019, Beni Mellal, Morocco

Said ADBI, Fatiha AIT YACINE, Abdellah AMINE and Belaid BOUIKHALENE, Sultan Moulay Slimane University, Beni Mellal, Morocco.

- Analysis and Design Process of a Decision-making System for the Moroccan University

The 5th International Conference on Business Intelligence, April 17 - 19, 2019, Beni Mellal, Morocco

Abdellah AMINE, Fatiha AIT YACINE, Rachid AIT DAOUD and Belaid BOUIKHALENE, Sultan Moulay Slimane University, Beni Mellal, Morocco.

En soumission :

- Realization of a Distributed “Intelligent-Schooling-Service», based on the Multi-Agent system and the Middleware JADE.

*Fatiha AIT YACINE, Belaid BOUIKHALENE, Abdellah AMINE, Sultan MoulaySlimane University, Morocco.

Iaescore Indonesian Journal of Electrical Engineering and Computer Science.

Chapitre I : Etat de l'art des Systèmes Multi-agents

I.1 Introduction

Les systèmes multi agents (SMA) et les agents autonomes fournissent une nouvelle méthode pour analyser, designer et implémenter des applications sophistiquées parce qu'ils font partie du domaine IAD (Intelligence Artificielle Distribuée) en bénéficiant aussi d'autres disciplines comme les sciences cognitives, sociologie, et psychologie sociale.

Aux débuts de l'intelligence artificielle distribuée (IAD), les recherches se sont concentrées sur la construction et le développement des systèmes illustrant la nouveauté du domaine. D'un point de vue théorique, les architectures individuelles des agents étaient au centre des préoccupations [6][7]. Ces dernières années, l'attention s'est lentement déplacée sur les aspects sociaux de la connaissance et de l'action [8] [9] [10][11] mettant l'accent sur l'approche systèmes multi-agents (SMA). Ceux-ci offrent deux points de vue pour la modélisation d'un système complexe :

- Agent : entité logicielle autonome capable de raisonner et d'interagir par échanges de connaissances [12] [13] avec les autres agents du système pour satisfaire ses propres buts ou les buts globaux du système ;
- Société : ensemble des représentations et des mécanismes nécessaires à la gestion des interactions et à l'organisation ou structuration des agents dans les systèmes.

L'objectif de cette partie est de faire le point sur les agents logiciels, les systèmes multi-agents ainsi que leurs architectures, et de comprendre comment ils arrivent à coopérer et à coordonnées leurs actions pour atteindre les objectifs assignés à leur organisation ou leur société tout en restant autonomes. On a abordé également les différents modes de communication entre les agents et les langages de communication les plus connus dans ce domaine. Ce qui va nous permettre de proposer plus loin aux chapitres (4 et 5) des architectures de modèles multi-agents pour les intégrer dans notre système d'information universitaire.

I.2 Agents et systèmes multi-agents

Les Systèmes Multi-Agents (SMA) constituent aujourd'hui une approche privilégiée pour la conception des systèmes complexes. Leur conception décentralisée est particulièrement adaptée aux applications des tables interactives qui nécessitent de faire interagir plusieurs entités. Généralement, un Système Multi-Agent est caractérisé selon cinq concepts présentés par Yves Demazeau[14] de la façon suivante : l'agent, l'environnement, les interactions, l'organisation et l'utilisateur.

Ainsi, un Système Multi-Agent est un système composé d'entités autonomes appelées agents. Un agent évolue dans un environnement et exécute un ensemble d'actions de manière autonome dans le but de satisfaire ses propres objectifs.

La définition d'un SMA s'impose. D'après Ferber[15] :

« On peut définir un système multi-agents comme un système composé des éléments suivants :

- Un environnement (par exemple un espace disposant d'une métrique) ;
- Un ensemble d'objets situés, c'est-à-dire auxquels on peut associer une position dans l'environnement ;
- Un ensemble d'agents, qui sont des objets particuliers (les agents diffèrent des autres objets par leur capacité à agir sur eux-mêmes et sur l'environnement) ;
- Un ensemble de relations, ou de contraintes, qui unit des objets et/ou des agents entre eux ;
- Un ensemble d'opérations déterminant les façons dont les agents peuvent agir sur les objets ;
- Un ensemble d'opérateurs représentant la façon dont ces actions sont effectuées et leur effet sur l'environnement. ».

En général la philosophie des SMA repose sur le principe de faire interagir plusieurs agents entre eux pour réaliser un objectif global. Chacun de ces agents détient une représentation partielle de son environnement, et possède des connaissances et des

compétences propres qui leur permettent d'évoluer dans l'environnement commun de manière à satisfaire leur but local.

La figure 2 illustre la structure typique d'un SMA.

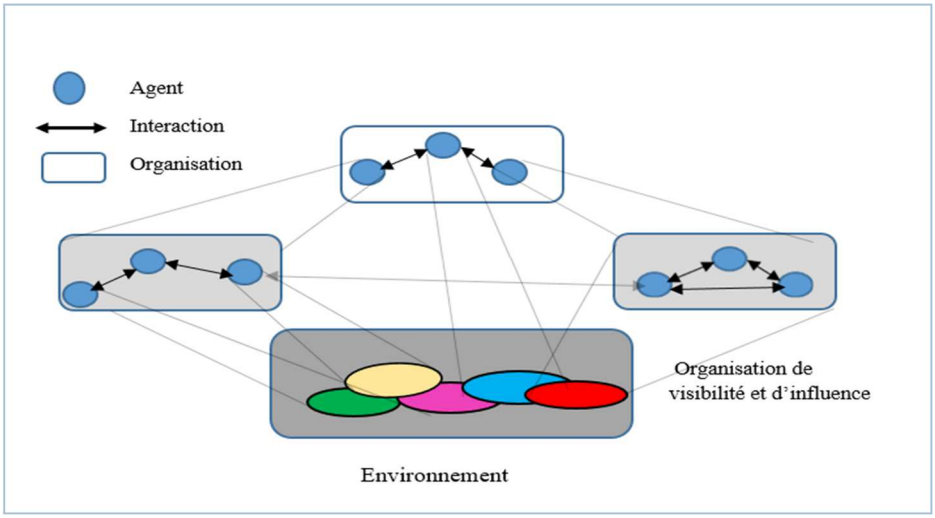


Figure 2: Structure typique d'un système multi-agents

Les agents du SMA sont regroupés en relations organisationnelles et ont chacun "une sphère d'influence", c'est-à-dire que chaque agent possède le contrôle d'une partie de l'environnement. Ces sphères d'influence peuvent parfois interférer et c'est ce qui engendre les situations d'interactions entre les agents des diverses organisations.

La force du paradigme multi-agents provient de la flexibilité et de la variété des types de ces interactions et des modèles d'organisation impliqués dans de tels systèmes. Par ailleurs, les SMA doivent avoir leurs succès à bien d'autres raisons, principalement du fait qu'ils répondent à certains besoins et qu'ils présentent certaines caractéristiques fonctionnelles qu'on ne retrouve pas forcément dans les autres technologies telles que la technologie orientée-objet.

I.3 Concept d'agent

I.3.1 Définition d'agent

Il n'existe pas de définition unifiée de ce qu'est un agent.

Nous présentons ici les définitions les plus pertinentes et les plus répandues :

- Un agent est un système informatique situé dans un environnement, capable de réaliser des actions flexibles et autonomes dans cet environnement, pour atteindre le but pour lequel il a été conçu [16].

- Un agent est une entité qui fonctionne continuellement et de manière autonome dans un environnement où d'autres processus se déroulent et d'autres agents existent [17].

- Un agent est plongé dans un environnement dont il a une perception partielle par des capteurs et sur lequel il peut agir via des effecteurs. Il possède donc une interface vers l'extérieur. Chaque agent est conçu pour résoudre une tâche particulière (but) participant à la résolution d'un objectif global du système. Un agent est autonome : il a le contrôle de son état interne et de son comportement. Il est capable d'adapter son comportement aux changements de l'environnement et de prendre des décisions qui permettront de satisfaire ses objectifs [18]. .

- Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.

On peut considérer comme définition générale :

- Un Agent est une entité
  - ✓ Qui agit d'une façon autonome ;
  - ✓ Pour atteindre les objectifs pour lesquels il a été conçu ;
  - ✓ Peut communiquer avec d'autres agents ;
  - ✓ Doté de capacités semblables aux êtres vivants.
- Un agent peut être processus, un robot, un être humain, etc.

I.3.2 Types d'agents

Nous distinguons deux grandes familles d'agents : les agents réactifs et les agents cognitifs [16].

Cette distinction tient essentiellement au processus décisionnel chez l'agent et à la représentation de l'environnement dont il dispose. Si l'agent est doté d'une représentation symbolique de l'environnement à partir duquel il est capable de formuler des raisonnements, nous disons qu'il est cognitif tandis que s'il ne dispose que d'une représentation limitée à ses perceptions, nous disons qu'il est réactif [15]. Qu'il soit réactif ou cognitif, un agent évolue toujours selon un cycle en trois étapes :

1. Perception : ses capteurs lui fournissent une vision locale de son environnement ;

2. Décision : suivant ses intentions, son état interne et sa perception, l'agent choisi une action à effectuer ;

3. Action : il modifie l'environnement par son action.

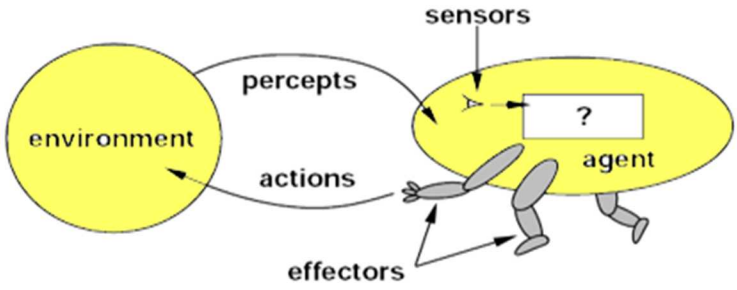


Figure 3: Etapes d'évolution d'un agent

Les types d'agents les plus couramment utilisés dans les SMA sont les agents cognitifs, les agents réactifs et les agents hybrides.

I.3.3 Agent Cognitifs

Agents cognitifs : Les agents cognitifs disposent d'une base de connaissances comprenant l'ensemble des informations disponibles et un raisonnement nécessaire à la réalisation de leurs tâches ainsi qu'à la gestion des interactions avec les autres agents

et avec l'environnement. De plus, ils possèdent des buts et des plans pour décider de leurs actions. Les groupes d'agents peuvent coordonner leurs activités et négocier entre eux pour résoudre leurs conflits. Ces comportements peuvent être assimilés à des comportements sociaux et les recherches dans ce domaine s'appuient sur les travaux de sociologie des organisations des groupes d'individus [15]. L'une des architectures cognitives les plus connues est l'architecture BDI : Belief (Croyance), Desire (Désir), Intention (Intention) [19] Les croyances correspondent aux informations dont dispose l'agent sur son environnement. Les désirs correspondent aux états de l'environnement que l'agent souhaiterait voir réalisés. Les intentions correspondent aux projets de l'agent pour satisfaire ses désirs.

Le modèle BDI a inspiré beaucoup d'architectures d'agents cognitifs [20]. Nous citons :

- PRS (Procedural Reasoning System) développée par Georgeff et Lansky en 1987 [21].
- dMARS (the distributed Multi-Agent Reasoning System) est une évolution de l'architecture PRS, réalisée par Woold ridge et al. en 1997 [22].
- Enfin, nous pouvons citer IRMA (Intelligent Resource-bounded machine Architecture) développée par Bratman et al. en 1988 [23], etc.

I.3.4 Agent Réactifs

Agents réactifs : Contrairement aux agents cognitifs, les agents réactifs ne sont pas nécessairement assez intelligents individuellement pour que le système ait un comportement global que l'on puisse qualifier d'intelligent [10]. Des mécanismes de réactions aux évènements et aux différentes perceptions, ne tenant pas compte ni d'une explication des buts, ni des mécanismes de planification, peut alors découler la résolution de problèmes complexes : le comportement complexe du système émerge alors de la coexistence et de la coopération des comportements simples de chaque agent.

Chaque individu pris séparément possède une représentation faible de l'environnement et n'a pas de buts. Mais ces derniers peuvent être capables d'actions évoluées et coordonnées. C'est le cas d'une société de fourmis, étudiée par Alexis Drogoul [24].

Agent Cognitif	Agent Réactif
Représentation explicite de l'environnement	Pas de représentation explicite
Peut tenir compte de son passé	Pas de mémoire de son historique
Agents complexes	Fonctionnement stimulus/réponse
Système composé de petit nombre d'agents	Système composé de grand nombre d'agents

Tableau 1: Les différences entre agents cognitifs et agents réactifs

I.3.5 Agent Hybrides

Agents hybrides : Les agents hybrides sont conçus pour combiner des capacités réactives à des capacités cognitives, ce qui leur permet d'adapter leur comportement en temps réel à l'évolution de l'environnement [25][26]. Dans le modèle hybride, un agent est composé de plusieurs couches, rangées selon une hiérarchie en trois couches [27] : la couche purement réactive, la couche intermédiaire et la couche sociale. Dans ces trois couches, on va de la simple capture d'informations à partir de senseurs jusqu'à la gestion de la coopération entre agents en passant par la construction d'abstraction à partir des connaissances.

Pour illustrer cette famille, nous pouvons citer l'architecture ASIC [28] utilisée pour le traitement numérique d'images, l'architecture ARCO [29] créée dans le cadre de la robotique collective et l'architecture ASTRO [30] développée pour être utilisée dans les systèmes multi-agents soumis à des contraintes de type temporel.

I.3.6 Agent Mobile

Un agent mobile est un agent capable de se déplacer d'un site physique à un autre. Ce type d'agent s'oppose aux agents statiques (ou stationnaires), qui s'exécutent seulement dans le système où ils ont commencé leur exécution. Les agents stationnaires communiquent avec leur environnement selon des moyens conventionnels tels que RPC (Remote Procedure Calling). Ces moyens peuvent être coûteux en termes de charge de

communication. Par contre, un agent mobile n'est pas lié au système dans lequel il débute son exécution [31]. Il est capable de se déplacer d'un hôte à un autre dans le réseau. Il peut transporter son état et son code d'un environnement vers un autre où il poursuit son exécution. Ceci a l'avantage de réduire le transfert des flux de données volumineuses dans le réseau : quand un grand volume de données est stocké dans un hôte éloigné, ces données seront traitées localement plutôt que transférées à travers le réseau. Il s'agit donc de déplacer les traitements vers les données plutôt que les données vers les traitements (figure 4).

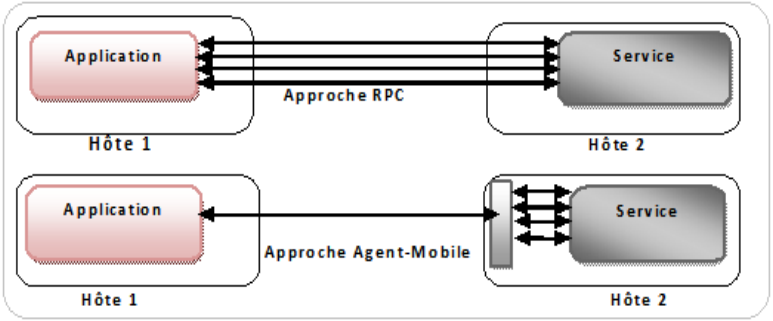


Figure 4: Réduction de la charge des réseaux

Un agent mobile peut avoir la même architecture qu'un agent réactif ou cognitif, bien qu'il y ait une nécessité d'avoir un module supérieur gérant la mobilité de l'agent.

I.4 Interaction entre agents

Une des principales propriétés de l'agent dans un SMA est celle d'interagir avec les autres agents. Ces interactions sont généralement définies comme toute forme d'action exécutée au sein du système d'agents et qui a pour effet de modifier le comportement d'un autre agent. Elles permettent aux agents de participer à la satisfaction d'un but global.

Ferber définit [15] les interactions comme étant la mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques.

Ces interactions prennent deux formes (directes et indirectes) :

- **Les interactions directes** sont en général des caractéristiques des agents cognitifs qui, travaillant pour un but précis, sont capables de communiquer intentionnellement. Ce type d'interaction se rapproche d'un acte de langage.

- **Les interactions indirectes** sont les caractéristiques des agents réactifs qui répondent à des stimulés de l'environnement par le dépôt d'information relative à des critères environnementaux [32].

Dans le cas des agents réactifs, Les interactions entre les agents sont modélisées comme suit :

1. **Coopération :** travailler ensemble dans un but commun, ceci est fait au travers de l'environnement sans aucune communication directe entre les agents ;
2. **Coordination :** organiser la solution d'un problème de telle sorte que les interactions nuisibles soient évitées ou que les interactions bénéfiques soient exploitées.

En général, les interactions sont mises en œuvre par un transfert d'informations entre agents ou entre l'environnement et les agents, soit par perception, soit par communication. Par la perception, les agents ont connaissance d'un changement de comportement d'un tiers au travers du milieu. Par la communication, un agent fait un acte délibéré de transfert d'informations vers un ou plusieurs autres agents. L'interaction peut être décomposée en trois phases non nécessairement séquentielles [33] :

- la réception d'informations ou la perception d'un changement,
- le raisonnement sur les autres agents à partir des informations acquises,
- une émission de message(s) ou plusieurs actions (plan d'actions) modifiant l'environnement. Cette phase est le résultat d'un raisonnement de l'agent sur son propre savoir-faire et celui des autres agents.

I.5 Concept d'Organisation des agents

Étant donné que les SMA peuvent être considérés comme une société d'agents coopérant ensemble pour accomplir collectivement un objectif donné, il est nécessaire de résoudre un problème d'organisation, généralement de façon dynamique.

On peut définir une organisation comme une structure décrivant comment les membres de l'organisation sont en relation et interagissent afin d'atteindre un but commun [34].

L'autonomie et le comportement proactif des agents constituant les SMA suggèrent que la conception de ces applications peut être réalisée en imitant le comportement et la structure des organisations humaines, car l'une des missions principales des SMA est de supporter et/ou de contrôler des organisations du monde réel. Un SMA peut, par exemple, aider à contrôler les activités du commerce électronique. Selon [35], la perspective organisationnelle conduit à une caractérisation générale d'un SMA :

Le système peut être décomposé en des sous-organisations distinctes. Un agent peut jouer un ou plusieurs rôles tout en coopérant et en respectant ses sous-organisations. Les interactions entre agents apparaissent via les capteurs et les émetteurs. Le rôle d'un agent détermine la portion de l'environnement dans laquelle il peut recevoir et émettre des données.

I.6 Environnement

L'environnement est l'endroit d'immersion dans lequel évoluent les agents. On distingue les plus souvent deux sortes : l'environnement social qui est composé des autres agents du système et l'environnement physique au sein duquel les agents évoluent. Un agent peut se placer dans l'un et/ou dans l'autre. L'environnement physique dépend largement de l'application traitée alors que l'environnement social dépend souvent des choix de conception des SMA. L'environnement est défini comme suit :

- L'environnement [36] est une première classe d'abstraction fournissant les conditions environnementales aux agents pour exister et sert d'intermédiaire pour les interactions entre les agents ainsi que pour l'accès aux ressources.

L'environnement peut être considéré comme la représentation du monde dans lequel les agents se situent [37]. L'environnement est modifiable par les agents, soit de façon globale, soit en faisant la distinction entre objets passifs (soumis aux actions des agents) et entités actives (les agents) [15].

I.7 Langage de communication entre agents

Grâce à la coordination un système multi-agents peut réaliser ses tâches avec plus d'efficacité qu'un seul agent. Mais pour coordonner l'activité d'un ensemble hétérogène

d'agents autonomes, il faut que les agents communiquent dans un langage compréhensible par tous les autres.

L'utilisation d'un langage commun implique que tous les agents comprennent son vocabulaire sous tous ses aspects concernant :

- **la syntaxe**, qui précise le mode de structuration des symboles ;
- **la pragmatique** pour pouvoir interpréter les symboles ;
- **l'ontologie** pour pouvoir utiliser les mêmes mots d'un vocabulaire commun.

Si on regarde l'évolution des langages de communication on constate une tendance à assurer un partage d'information de plus en plus complexe. Au début on peut remarquer que le partage concernait les objets (Remote Method Invocation, CORBA) ; en suite il a concerné les des connaissances (faits, règles, procédures de traitement des connaissances). Les ACL du présent concernent le partage des états mentaux (croyances, désirs, intentions). On peut citer ici les langages KQML et FIPA-ACL.

Tous les deux (FIPA et KQML) sont fondés sur la théorie Acte de langage avancée par Searle en 1960 et augmentée par Winograd et Flores dans les années (70). Pour inciter des agents à se comprendre ils doivent non seulement parler la même langue, mais avoir une ontologie commune. Les messages sont des actions ou des actes communicatifs, car ils sont prévus pour effectuer une certaine action en vertu de l'envoi.
Ayant une syntaxe similaire à KQML le langage s'appuie sur la définition de deux ensembles :

- un ensemble d'actes de communication primitifs, auquel s'ajoutent les autres actes de communication pouvant être obtenus par la composition de ces actes de base
- un ensemble de messages prédéfinis que tous les agents peuvent comprendre

Le langage FIP A-ACL suit le style de KQML (utilisant des performatifs issus de la théorie des actes de langage et quelques paramètres complémentaires), mais avec une sémantique mieux spécifiée.

I.8 Types d'application des systèmes Multi-agents

Le paradigme orienté agent trouve son utilité en s'attaquant à la modélisation des systèmes ouverts à l'extensibilité, flexibles et auto-adaptables. En effet, le paradigme orienté agent cherche à interconnecter des ressources logicielles de façon à créer un système intégré permettant de résoudre des problèmes complexes et de s'auto-adapter à différentes situations.

Généralement, les types d'application des SMA sont décomposés en trois grandes classes [38]:

- la modélisation des phénomènes du monde réel afin de les simuler pour mieux comprendre leurs comportements. Ceci est réalisé par l'observation, la compréhension et l'explication de leurs comportements et de leurs évolutions. Ce sont par exemple, des applications de simulation de phénomènes sociaux, environnementaux, éthologiques, etc.

- la conception des programmes dans lesquels des agents qualifiés d'assistants jouent le rôle des êtres humains. La notion d'agent simplifie la conception de ces programmes et amène de nouvelles problématiques centrées utilisateurs telles que la communication, la sécurité, etc. On trouve un exemple de telle approche dans les systèmes de ventes aux enchères dans lesquels les agents jouent les rôles de commissaires-priseurs et d'acheteurs.

- la résolution distribuée des problèmes : il s'agit de concevoir des systèmes artificiels intégrant un ensemble d'agents et une dynamique organisationnelle permettant l'accomplissement collectif d'une tâche. L'hypothèse de base est que des agents, par leurs comportements et leurs interactions locales, peuvent donner lieu, à un niveau global, à un comportement d'ensemble cohérent et fonctionnel. Ce sont par exemple les fourmis qui partent de leur fourmilière à la recherche de la nourriture en créant le plus court chemin et qui coopèrent pour réaliser cette tâche. C'est plutôt ce type d'application qui nous concerne.

Ainsi, les systèmes multi-agents peuvent être utiles dans plusieurs contextes.

D’ailleurs, certains systèmes industriels tels que la gestion des ressources hydrauliques [39], le contrôle du trafic aérien [40] et les télécommunications [41] ont appliqué ce paradigme.

I.9 Conclusion

Dans ce chapitre, nous avons défini les principaux concepts des agents et des SMA utilisés tout au long de ce mémoire. Ces définitions vont permettre au lecteur de comprendre le sens précis dans lequel nous les avons utilisés.

Chapitre II : Paradigmes et Méthodologies

II.1 Introduction

Nous présentons brièvement dans ce qui suit l'évolution des paradigmes de développement informatique ainsi que quelques techniques et méthodologies qui sont appropriées à ces paradigmes.
Dans le contexte informatique, un paradigme de développement est une manière de représenter le monde dans le but de concevoir un programme informatique. Depuis l'apparition de l'informatique, plusieurs paradigmes ont été adoptés (paradigme procédural, paradigme orienté objet, etc.). Chaque paradigme identifie une classe de problèmes et fournit des techniques permettant de traiter ces problèmes.

Différents paradigmes de développement ont vu le jour, introduisant de nouveaux concepts permettant de structurer les programmes informatiques. Les paradigmes de développement ainsi que les concepts qu'ils introduisent permettent d'augmenter le niveau d'abstraction afin de faciliter le développement. Le passage d'un paradigme à un autre est justifié par la nécessité d'encapsuler certains mécanismes de bas niveau rendant la tâche de développement moins complexe. Ainsi, les différents paradigmes de développement sont en fait complémentaires et s'inscrivent dans une logique de continuité.

Les techniques d'ingénierie (Méthodologie) englobent les différents moyens mis à la disposition des développeurs afin de faciliter l'application des paradigmes. Une bonne application des paradigmes est conditionnée par le développement de techniques d'ingénierie appropriées fournissant des cadres conceptuels, technologiques et méthodologiques qui guident le développeur lors des activités de développement. Le développement d'une application informatique quelle qu'elle soit nécessite l'application de différentes activités allant de la collecte et l'analyse des besoins jusqu'au déploiement et la maintenance du système.
Pour guider les développeurs lors de ces activités, des méthodologies sont proposées. Booch définie une méthodologie comme "un ensemble de méthodes appliquées tout au long du cycle de développement d'un logiciel, ces méthodes étant unifiées par une

certaine approche philosophique générale"[42]. L'approche philosophique générale est définie par les paradigmes de développement. Les méthodes quant à elles définissent la manière d'appliquer cette approche en fournissant des notations et des modèles permettant d'exprimer les spécifications relatives à chaque activité (modèle des besoins durant l'analyse, modèle architectural lors de la conception, code source lors de l'activité d'implémentation etc.).

II.2 Paradigme procédural

Le paradigme procédural peut être considéré comme le paradigme de développement le plus ancien. Il a permis de s'affranchir du code machine et de renforcer une séparation entre la représentation des données et les traitements faits sur ces données. La programmation procédurale est basée sur le concept d'appel procédural. Une procédure, aussi appelée routine, sous routine, méthode ou fonction. N'importe quelle procédure peut être appelée à n'importe quelle étape de l'exécution du programme. Une réutilisation des sous-routines (procédures) a été obtenue. Cependant, le développement des systèmes complexes utilisant ce paradigme a généré des structures trop complexes. En effet, l'interopérabilité entre les procédures se fait par les variables globales qui sont accessibles par différentes procédures. Lors de la maintenance, la modification d'une procédure peut avoir des conséquences inattendues sur le résultat d'autres fonctions ou procédures liées par des variables globales. Ainsi, la maintenance des systèmes développés avec le paradigme procédural est difficile et coûteuse et la réutilisation du code se trouve souvent compromise.

Les concepteurs et les développeurs se sont rapidement aperçus que la programmation procédurale ne répondait plus aux exigences du marché en matière d'extensibilité, de fiabilité, de testabilité, de maintenance, de réutilisabilité et autres standards de qualité. Un paradigme de développement plus modulaire est devenu nécessaire. C'est ainsi que le paradigme orienté objet a vu le jour.

II.3 Paradigme objet

La programmation Orienté Objet, répond mieux aux critères demandés par les entreprises. Le paradigme orienté objet a permis une meilleure protection des données grâce à l'encapsulation des données. Cette programmation permet le développement modulaire, une bonne réutilisabilité des composants (ce qui est très important au niveau des coûts de développement), une extension des systèmes grâce à l'héritage, une plus grande facilité de maintenance, etc. Mais devant la complexité sans cesse croissante des systèmes informatiques actuels, le paradigme orienté objet n'a pas réussi à limiter les difficultés de développement inhérentes aux systèmes distribués et concurrentiels compte tenu de sa faible granularité.

II.4 Paradigme composant

Le passage au paradigme orienté composant a permis d'augmenter le niveau d'abstraction et de changer la manière de développer les systèmes informatiques. Un composant est décrit comme une brique logicielle composable et facilement réutilisable. IL n'existe pas de réel consensus pour définir le composant. Pour Szperski [43], un composant est ainsi « une unité de composition dont les interfaces et les dépendances contextuelles sont spécifiées sous forme de contrats explicites. Il peut être déployé indépendamment et peut être composable par des autres composant ».

Pour Heinman et Council [44], un composant « est un élément logiciel conforme à un modèle appelé modèle de composants qui définit des standards de composition et d'interaction ». Un modèle de composant est habituellement indépendant de tout langage d'implémentation.

D'une manière simple nous pouvons considérer un composant schématisé dans la figure suivante comme une unité réutilisable pour la composition et le déploiement, décrivant ses services requis, ses services fournis et ses paramètres.

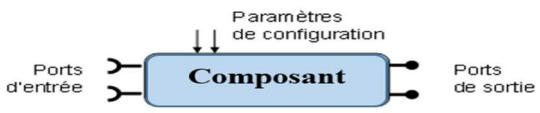


Figure 5: Représentation graphique d'un composant

L'apport des composants par rapport à l'objet est notamment une plus grande modularité et facilité de réutilisation grâce à une encapsulation plus forte (notion de boite noire). Mais la composition des composants est encore rigide et complexe du fait qu'elle est basée uniquement sur les protocoles d'interaction et les flux d'entrée et de sortie des composants.

Il existe plusieurs solutions industrielles à base de composants qui émergent aujourd'hui à savoir la (COM, JavaBeans, EJB, …..). La programmation orientée composant se base sur les connections.

II.5 Paradigme orienté service

Pour alléger la gestion des interopérabilités et réduire le couplage entre les composants, le paradigme orientée service a été proposé. Ce paradigme propose de découper les fonctionnalités d'une application en services métier, réutilisables dans d'autres applications. Un service métier peut être défini comme une fonction de haut niveau. Les opérations proposées par cette fonction encapsulent plusieurs autres fonctions et opèrent sur un périmètre de données large. Ainsi, les services ont un niveau de granularité plus élevé que les composants. La description d'un service est complètement indépendante des plates-formes et outils.
Dans le cas des services web, un service n'est autre qu'une application exposée par le biais d'une interface XML, connue sous le nom de services Web. Les services Web sont des couches d'invocation compréhensibles potentiellement par l'ensemble des systèmes. Les services Web interagissent les uns avec les autres par le biais d'envoi de messages. De ce fait, la modélisation de services Web met en jeu la description d'interactions et de leurs interdépendances, aussi bien d'un point de vue structurel (types de messages échangés) que d'un point de vue comportemental (flot de contrôle entre interactions).

Trois types de modèles ont été distingués pour décrire les interactions entre les services :

- la chorégraphie : il s'agit de décrire les interactions entre les services Web sans avoir de coordinateur central,

- l'interface : il s'agit de décrire les actions d'envoi et de réception de message pour chaque service. Une interface permet de spécifier quelles sont les données nécessaires à l'exécution du service, et ce qu'il fournit en retour).
- L'orchestration : il s'agit de décrire un processus automatique d'organisation, de coordination, et de gestion des services Web.

Les applications développées selon le paradigme orienté services présentent de nombreux avantages parmi lesquels la flexibilité et la réactivité.

Cependant, certains obstacles ne sont pas encore surmontés. Le chantier de standardisation relatif à l'orchestration et la chorégraphie est loin d'être achevé ce qui pose un problème d'interopérabilité. La mise en œuvre d'applications orientées service souffre d'un manque d'outils méthodologiques. Enfin, les services ne sont pas conscients de leur environnement et ne peuvent réagir en cas d'imprévus.

II.6 Position du paradigme orienté agent

Le paradigme orienté agent est né dans les années 80, mais reste encore d'actualité puisqu'il propose des mécanismes d'abstraction qui permettraient probablement de résoudre plusieurs problématiques rencontrées actuellement. En effet, les recherches menées autour de ce paradigme élaborent des mécanismes pour la construction de systèmes complexes évoluant dans un environnement dynamique et ouvert (c'est-à-dire dont la cardinalité et la topologie des composants le constituant peut être variable). Ces systèmes complexes comportent un grand nombre de composantes en interaction. L'organisation de ces composantes peut être à priori inconnue. Le contrôle de tels systèmes ne peut pas se faire de manière centralisée.

Pour traiter ce type de système, le paradigme orienté agent propose une nouvelle philosophie de développement qui va au-delà d'une simple décomposition conceptuelle ou fonctionnelle comme il se fait avec les paradigmes de développement actuels. Suivant le paradigme orienté agent, les composantes du système appelées agents.

II.7 Analyse comparative de quelques paradigmes

Dans cette section, nous allons faire une analyse comparative des quelques paradigmes. Les paradigmes informatiques que nous analysons ici sont, les objets, les composants, les services et les agents. Cette analyse est restrictive au côté structurel, qui est identifié par l'ensemble des concepts tangibles au niveau du modèle, nous utilisons comme paramètres, la hiérarchisation et le couplage structurel des entités.

Pour pouvoir mieux comparer les concepts des paradigmes, objet, composant, service et agent, nous avons choisi dans un premier analyse de les replacer dans une perspective générale d'évolution de la programmation ce qui permet de dire que le concept de services agrège celui de composants, agrégeant lui-même celui d'objets .

La figure 6 illustre cette structure hiérarchique à 3 niveaux.

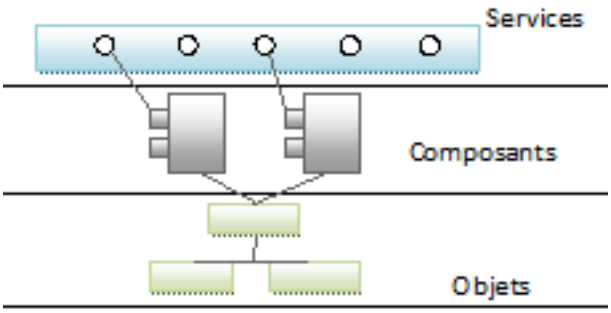


Figure 6: Architecture hiérarchique à objets/composant/services

Les agents ne fonts pas partie de cette première partie de comparaison du fait que leur histoire de naissance et d'évolution est différent.

Dans cette deuxième partie de comparaison, nous choisissons un cadre d'analyse à deux dimensions. Ce cadre est un repère dont la première dimension représente le dynamisme (flexibilité). La projection sur la dimension du niveau de couplage entre différentes entités.

La flexibilité du couplage. Elle représente la capacité de mettre en relation plusieurs entités logicielles. L'évolution de la programmation montre le besoin croissant de représenter et manipuler ces relations de manière indépendante de la description des entités elles-mêmes, pour offrir une vision architecturale indépendante de l'implantation.

Les composants logiciels apportent une amélioration pour le couplage structurel par L'externalisation des références et par leurs descriptions explicites sous la forme d'interfaces fournies (de sortie) et d'interfaces requises (d'entrée). La manipulation du couplage devient ainsi explicite et externe aux entités logicielles. Le couplage est explicité par des connecteurs, qui vont relier les interfaces des composants. Les identifiants de ces interfaces sont appelés des « ports », d'entrées ou de sorties. Dans le paradigme objet, un objet référence un autre par avoir un attribut dont la valeur est l'identifiant de l'objet référencé ce que implique des modifications à l'intérieur de l'objet en cas de besoin de modification de liaisons (couplage fort). Pour les composants la reconfiguration souhaitée s'opère par un simple ajout de connecteur.

Le concept d'architecture logicielle, assemblage de composants via des connecteurs bien explicites, représente de ce fait une avancée majeure. Le concept de service apporte une dynamicité (découverte de services) et un contrôle par l'entité elle-même (sélection et invocation de service). Les systèmes multi-agents poussent encore plus loin cette organisation du couplage et de la collaboration (coordination, décomposition, négociation. . .) à l'aide de connaissances (organisations, tâches, plans, protocoles. . .). La figure 7 illustre cette analyse en présentant le dynamisme (flexibilité) en fonction du couplage des paradigmes étudie.

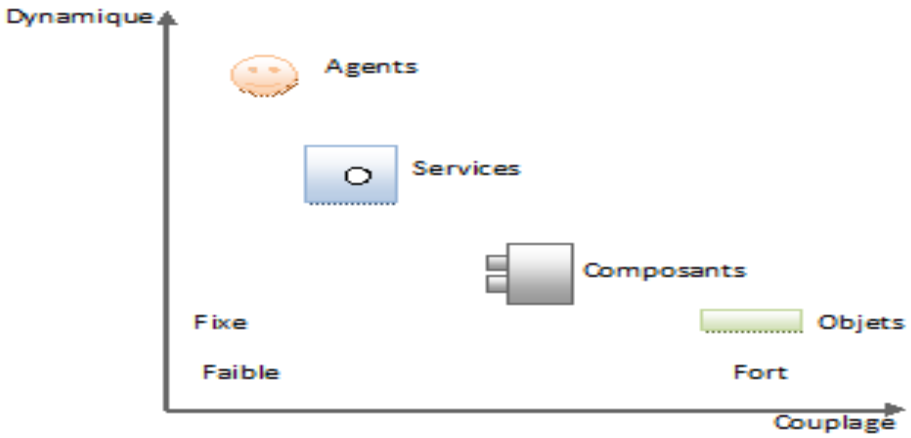


Figure 7 : Dynamisme en fonction du couplage des paradigmes

II.8 Evolution des Méthodologies relatives aux paradigmes

II.8.1 Introduction

Dans cette partie nous présentons brièvement l'évolution des méthodologies et des techniques relatives à l'évolution des paradigmes.
L'évolution des techniques d'ingénierie a suivi celle des paradigmes de développement. Avec l'augmentation des niveaux d'abstraction traités, les techniques d'ingénierie devaient s'adapter à de nouveaux besoins. Dans les années 80, le développement était principalement considéré sous l'angle de la programmation. Il était alors effectué en interne par une seule personne ou un petit groupe de personnes. On parlait de "programming in the small" [45]. Actuellement, les systèmes sont amenés à devenir des systèmes ouverts et coopératifs, en constante interaction les uns avec les autres.
Nous sommes donc passés à l'ère du "programming in the large"[45] où le développement est considéré à large échelle c'est-à-dire effectué par plusieurs équipes, parfois géographiquement éloignées et faisant coopérer plusieurs applications distribuées. Ce type de développement ne peut plus se concentrer uniquement sur la phase d'implémentation. De nouveaux outils deviennent alors nécessaires pour faciliter le travail de développement.

II.8.2 Méthodologies et techniques classiques

Plusieurs environnements de développement intégrés (EDI) ont vu le jour pour répondre aux besoins du paradigme procédural. Ces environnements de développement aidaient les programmeurs à structurer leurs applications lors de la phase d'implémentation. La première génération d'EDI ne supportait qu'un seul langage à la fois. Avec les mutations technologiques des années90 (application de plus en plus étendue du paradigme orienté objet), plusieurs ajouts fonctionnels améliorant la productivité des développeurs ont été proposés. Ainsi, les environnements de développement intégrés ne sont plus dédiés à un seul langage mais à un ensemble de langages. C'est certainement, l'environnement Eclipse qui a le plus marqué cette mutation. Son but étant de fournir une plate-forme modulaire pour permettre le développement informatique de systèmes complexes et de grande taille. Cet

environnement, écrit en Java, est extensible, universel et permettant de créer des projets mettant en œuvre n'importe quel langage de programmation.

Malgré l'efficacité des environnements de développement de deuxième génération ainsi que leurs différentes possibilités, leur aide reste limitée, dans la mesure où ils n'ont pas la capacité de couvrir la totalité du cycle de développement. En effet, ils restent pour la plupart dédiés à la phase d'implémentation. Une mutation actuelle des environnements de développement vise à intégrer les autres phases de développement telles que l'analyse, la conception et la validation des systèmes. Le principal but recherché est la génération automatique du code à partir de modèles conceptuels vérifiés et validés. Pour atteindre ce but, les EDI devraient permettre de couvrir la totalité du cycle de développement ce qui n'est pas actuellement le cas. Les ateliers de génie logiciel (AGL) sont apparus dans les années 80 pour mieux structurer le travail de développement de logiciels durant les phases préliminaires de développement (principalement l'analyse et la conception). Ils intègrent un ensemble d'outils permettant de saisir les spécifications relatives au système en utilisant un formalisme graphique ou textuel.

L'utilisation des AGLs a connu un grand engouement avec l'apparition du paradigme orienté objet et surtout l'introduction du formalisme UML (Unified Modelling Language) [46] qui est un langage unifié de spécifications des applications orientées objet.

II.8.3 Méthodologies du paradigme composant

L'utilisation du paradigme orienté composant nécessite la spécification du système à un niveau d'abstraction plus élevé que les concepts objet. Un raisonnement approfondi sur les composants à intégrer dans le système, la manière de les assembler et les contraintes qui doivent être respectées durant son exécution ainsi que celles qui doivent être conservées lors de son évolution est nécessaire. C'est l'architecture logicielle qui permet de représenter ces différents aspects.

D'après la définition fournie par d'ANSI/IEEE [47], une architecture logicielle ne se contente pas de décrire uniquement l'organisation des composants du système en précisant leurs relations ; l'architecture doit être un cadre pour la satisfaction du cahier des charges. Elle est la base technique pour la conception comme la base gestionnaire pour l'estimation des coûts et du processus de gestion. Au début des années 90 on a

assisté à l'établissement d'un domaine consacré au développement centré architecture [48] [49]. Dans ce domaine, les recherches sont axées sur l'étude des langages et formalismes nécessaires à la représentation et la documentation des architectures logicielles.

Actuellement, il n'existe pas de notation unique standardisée permettant de représenter les architectures logicielles. Certains auteurs voient en UML un formalisme pertinent permettant de décrire différentes vues du système [50]. Par exemple, Medvidovic et al. [51] proposent différentes manières de modéliser les architectures logicielles en UML (version 1.5). Il' est avéré, à la suite de cette étude, que l'utilisation d'UML tel quel devenait assez complexe, du fait qu'UML ne fournissait pas de moyens simples permettant de décrire les composants, qui avaient un niveau d'abstraction plus élevé que les objets. Il a été alors nécessaire d'étendre UML mais les outils fournis par les environnements basés sur UML ne permettent pas de gérer la description d'architectures logicielles.

D'autres types de formalismes ont vu le jour ; ce sont des Langages de Description d'Architecture (appelés en anglais Architecture Description Langages ou ADL). Nous allons ici adopter cette appellation anglaise ADL, fréquemment utilisée dans la littérature. Un ADL fournit les abstractions selon lesquelles les architectures vont être décrites. Généralement, il décrit une architecture logicielle sous la forme de composants, de connecteurs et de configurations [52].Les composants représentent les unités de calcul et de stockage de données dans un système logiciel. L'interaction entre ces composants est encapsulée par les connecteurs. Dans la configuration, l'architecte instancie un nombre de composants et un nombre de connecteurs. Il les lie entre eux afin de construire son système.
Les ADL se focalisent sur des aspects particuliers de la conception d'architectures logicielles. Par exemple, Darwin [53] est plus axé sur la modélisation d'architectures distribuées et la prise en compte de leur reconfiguration dynamique. WRIGHT [54] s'intéresse à la formalisation des connexions entre composants, tandis que Acme [55] ou ArchWare représentent des exemples d'ADL à objectif général, c'est-à-dire offrant des abstractions génériques permettant de décrire différentes sortes d'architectures. En se basant sur les avancées faites au niveau des ADL, la nouvelle version 2.0 du langage UML apporte des avancées significatives à la modélisation des systèmes à base de

composants. Dans cette nouvelle version, un composant peut être modélisé à tous les niveaux du cycle de développement.

II.8.4 Méthodologies dirigées par les modèles

Comme nous l'avons vue pour les autres paradigmes de développement, chaque paradigme a été accompagné par des méthodologies de spécification permettant de représenter et de raisonner sur le système de manière abstraite, une Ingénierie Dirigée par les Modèles (IDM) ou Model Driven Engineering (MDE) en anglais à vue le jour. Cette méthodologie vise à proposer une approche unificatrice basée sur l'utilisation systématique des modèles tout au long du cycle de développement. La nouveauté dans l'approche IDM consiste à passer des modèles purement contemplatifs aux modèles productifs. En effet les autres techniques traditionnelles étaient produites dans le but de documenter l'application en cours de développement afin de faciliter la compréhension et la communication entre les différents participants au projet de développement et ils étaient délaissés dès que l'activité de codage commençât.

IDM vise à fournir un cadre conceptuel, technologique et méthodologique dans lequel les modèles sont au centre des activités de développement.
Comme le souligne Jean-Marc Jézéquel [56], chaque processus de développement, quel que soit son type, comporte un certain nombre de phases (comme l'expression des besoins, l'analyse, la conception, l'implantation ou encore la validation). Dans chaque phase un certain nombre de modèles (appelés aussi artefacts) sont produits qui peuvent prendre la forme de documents, de diagrammes, de codes sources, de fichiers de configuration, etc.
Ces modèles qui représentent les différentes vues du système, sont souvent produits de manière indépendante. Il est alors important de s'assurer de la cohérence entre ces différentes vues. Ceci constitue le cheval de bataille du domaine de l'ingénierie dirigée par les modèles.
L'utilisation de ces modèles permet de capitaliser les connaissances et le savoir-faire à différents niveaux d'abstraction qu'elles soient des connaissances du domaine métier ou du domaine technique. On couvre ainsi les différentes vues du système.

II.9 Méthodologies de modélisation des SMA

II.9.1 Introduction

Au regard des caractéristiques d'agent, il apparaît que l'approche orientée-agent dans le développement de logiciel consiste en une décomposition du problème en agents ayant des interactions, une autonomie, et un objectif spécifique à atteindre. Les concepts clés d'abstraction liés à un système orienté-agent sont : agent, interaction, organisation.

Bien qu'il existe une similarité superficielle entre objet et agent, la terminologie objet n'est pas adaptée aux systèmes agents :

- ✓ les objets sont généralement passifs alors que les agents sont permanents actifs
- ✓ les agents sont autonomes et responsables de leurs actions alors que les objets n'en sont toujours pas ;
- ✓ on ne peut prévoir tous les comportements des agents dans les systèmes ;
- ✓ l'approche orientée-objet ne fournit pas un ensemble adéquat de concepts et de mécanismes pour modéliser les systèmes complexes dans lesquels les rapports évoluent dynamiquement ;
- ✓ certains chercheurs définissent un agent comme un objet actif ayant une autonomie.

Les méthodes de développement orientées objet, au vu des différences entre les objets et les agents, ne sont pas directement applicables au développement de SMA. Il est alors devenu nécessaire d'étendre ou de développer de nouveaux modèles, de nouvelles méthodologies et de nouveaux outils adaptés au concept d'agent.

II.9.2 Quelques méthodologies de modélisation des SMA

En général il y a deux grands groupes de méthodologies pour le développement du logiciel orienté agent. Le premier groupe étend ou adapte les méthodologies orientées objet pour prendre en compte les caractéristiques des agents, alors que le deuxième groupe part des méthodologies de l'ingénierie de connaissances.

Actuellement, il existe un nombre important de méthodologies et d'outils de conception de SMA. Chacune d'entre elles se fonde sur une base conceptuelle propre et recouvre un certain nombre d'aspects dont la prise en compte est considérée essentielle. La plupart de ces propositions méthodologiques sont inspirées des résultats du domaine du génie logiciel orienté objets. Dans cette section nous présentons un certain exemple de ces méthodologies :

AUML a pour but d'étendre UML avec des facilités permettant de décrire des agents. Ce formalisme a comme but de recommander une technologie pour l'adoption d'une sémantique, d'un méta-modèle et d'une syntaxe abstraite communes pour l'analyse et la conception des méthodologies basées sur les agents. Cette technologie devra permettre de couvrir le cycle de vie des produits et des outils de travail AUML et être en accord avec les spécifications existantes de FIPA et l'OMG (Object Management Group).

M-UML a été proposée par K. Saleh et C. El-Morr (2004) [57] pour modéliser les agents mobiles.

GAIA est une méthodologie où le système multi-agent est vu comme une organisation composée de rôles interagissant entre eux. La méthodologie prend comme point de départ une description de la structure organisationnelle du système, composée de rôles et d'interactions entre eux. Cette méthodologie aborde la conception d'un SMA sans un modèle d'architecture d'agent préétabli.

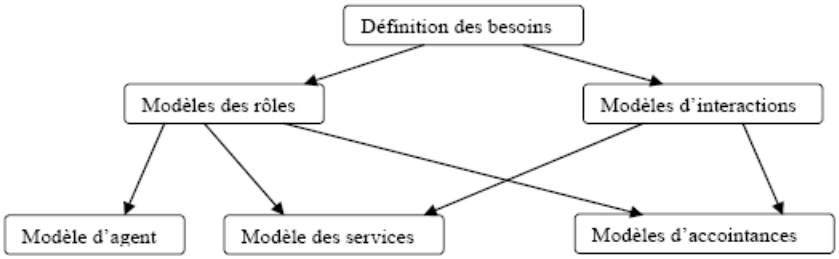


Figure 8: Modèle d'architecture d'agent préétabli.

AALAADIN est un méta-modèle reposant sur les concepts organisationnels que sont les agents, groupes et rôles (AGR) L'organisation est définie comme un cadre d'activité et d'interaction, mis en place par la définition de groupes, rôles et de leurs relations, et

considérée comme l'ensemble des relations structurelles dans un ensemble d'agents. Un SMA est vu par l'approche AALAADIN, comme un ensemble de groupes d'agents interconnectés par l'intermédiaire d'agents appartenant à plusieurs groupes. Aucune contrainte ou pré requis n'est imposée sur l'architecture interne des agents et aucun formalisme où modèle particulier n'est supposé pour en décrire leur comportement. Chaque agent est simplement décrit comme une entité autonome communicante qui joue des rôles au sein des différents groupes. La méthode AALAADIN a donné lieu au développement de la plate-forme MadKit.

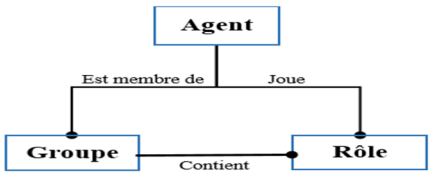


Figure 9:L'approche AALAADIN

Afin de donner un cadre méthodologique au processus de modélisation, nous allons par la suite appliquer le formalisme AUML et par conséquent UML 2.1 qui contient un grand nombre de concepts liés aux agents. En fait, UML est un standard largement connu, accepté et a fait ses preuves pour la modélisation des applications orientées objet. Le fait que le paradigme des agents est considéré comme une extension du paradigme objet a conduit plusieurs auteurs pour étendre UML afin de tenir compte des concepts agents.

II.10 Normes

L'émergence des nombreuses plateformes expérimentales a rendu essentielle la proposition d'une harmonisation grâce à la standardisation des différents concepts communs pouvant être identifiés. Cette normalisation devrait permettre à terme de rendre compatibles les différents systèmes.

MASIF : la norme MASIF a été spécifiée par l'Object Management Group (OMG) qui se préoccupe généralement de l'hétérogénéité entre les systèmes, comme dans le cas de

CORBA. Dans cette optique, le but, dans la norme MASIF est de décrire les notions élémentaires permettant l'échange des agents entre différentes plateformes. Pour ce faire, elle standardise la manière de gérer le code des agents, leur identification, la migration et l'adressage local.

FIPA : en revanche, la communauté à l'origine de FIPA étant celle des systèmes multi agents, plus proche de l'intelligence artificielle, elle va se situer à un niveau plus élevé, c'est-à-dire le niveau applicatif en décrivant les éléments nécessaires à la réalisation d'une application et principalement en détaillant la communication entre les agents. Le but est de décrire un ACL (Agents Communication Language), les ontologies et les protocoles de négociation permettant ainsi de définir parfaitement les interactions entre les agents.

Les deux standards MASIF et FIPA sont considérés comme des religions de la communauté de logiciels d'agents. Leurs différences résident dans le fait que MASIF vise à permettre aux agents mobiles de migrer entre les systèmes d'agents du même profil par l'intermédiaire des interfaces normalisées de CORBA, contrairement à FIPA qui permet l'interopérabilité d'agents intelligents par l'intermédiaire de la communication normalisée des agents et des langues.

Ces deux normes tendent plus vers la complémentarité que vers la divergence. C'est déjà le cas de FIPA qui a inscrit dans son planning l'intégration des règles de MASIF sur la gestion de la migration.

II.11 Plateformes des SMA

Comme on a vu pour les méthodologies, il existe à l'heure actuelle une multitude de plateformes multi-agents dédiées à différents modèles d'agent. Les plateformes fournissent une couche d'abstraction permettant de facilement implémenter les concepts des systèmes multi-agents. D'un autre côté, ces plateformes permettent aussi le déploiement de ces systèmes. Ainsi, elles constituent un réceptacle au sein duquel les agents peuvent s'exécuter et évoluer. En effet, les plateformes sont un environnement

permettant de gérer le cycle de vie des agents et dans lesquelles les agents ont accès à certains services.

Il existe à l'heure actuelle plusieurs plateformes d'agents, facilitant le développement et la manipulation des systèmes multi agent. Le choix d'une plateforme d'agent a une grande influence sur la conception et la mise en œuvre des agents, FIPA a défini les normes nécessaires qu'une plateforme d'agent devrait respecter. Ces normes existent pour assurer une conception uniforme des agents indépendamment de la plateforme.

Dans ce qui suit, nous présentons quelques plateformes. Cette liste représente cependant celles les plus utilisées et les plus citées dans la littérature. Nous décrivons et comparons brièvement les suivantes :

MADKIT :(Multi-Agent Development Kit) a été développée en 1998 au LIRMM (Montpellier) par Olivier Gutchneck[58], est une plateforme des agents flexible, capable de supporter plusieurs modèles de communication simultanément. Mais Bien qu'elle puisse supporter le développement de divers systèmes, elle semble bien adaptée pour les applications de simulation.

GAMA :(Gis and Agent-based Modelling Architecture) est une plateforme générique pour la modélisation et la simulation orientée agent. Le développement de cette plateforme est en parallèle avec l'implémentation de plusieurs modèles complexes qui appartiennent à différents domaines. La diversité des modèles développés nous aide à vérifier la généralité de la plateforme et l'assurer d'une part, et à découvrir les fonctionnalités manquantes de la plateforme d'autre part.

JADEX [59]: est une plateforme multi-agents développée en JAVA par l'université de Hambourg, compatible avec de nombreux standards et capable de développer des agents suivant le modèle BDI (Croyance Désire Intention) du fait qu'elle prend en compte l'architecture des agents hybrides (c'est-à-dire des agents qui sont à la fois réactifs et proactifs)

JADE :(Java Agent DEvelopment) développée par Bellifemmine [60] est un Framework de développement de systèmes multi-agents, open-source et basé sur le langage Java. Il offre en particulier un support avancé de la norme FIPA-ACL, ainsi que des outils de validation syntaxique des messages entre agents basés sur les ontologies.

Cette plateforme sera plus détaillée dans ce qui suit, du fait qu'elle répond au besoin des modèles SMA que nous allons réaliser dans les chapitres 4 et 5.

Le tableau ci-dessous présente une comparaison de ces plateformes.

Caractéristiques	**GAMA**	**JADEX**	**MADKIT**	**JADE**
Etat	Moyenne Disponibilité	Logiciel libre	Logiciel libre	Logiciel libre
Documentation		Moyenne disponibilité	Moyenne disponibilité	Haute disponibilité
Conformité FIPA	Conforme	Conforme	Non conforme	Conforme
Langage de programmation	JAVA	JAVA, KQML, XML	GAML	JAVA, XML

Tableau 2: Comparaison entre les différentes plateformes

II.12 Conclusion

Dans ce chapitre nous avons décrit de nombreuses méthodes orientées agent et objet ainsi que quelques outils supportant les méthodologies SMA. La description de ces méthodes permettra au lecteur d'en avoir une idée sur les outils de spécification, de modélisation et de conception.

De nombreuses méthodologies orientées agent existent actuellement, chacune rendant compte de sa propre vision des concepts agents et multi-agents.

Chapitre III : Analyse et réalisation d'un modèle distribué à base du paradigme objet

III.1 Introduction

Les chapitres suivants seront consacrés à l'exploitation de l'étude comparative, aboutis dans les chapitres précédents pour résoudre la problématique de notre travail.
Dans un premier temps, nous allons faire la conception et la mise en œuvre d'une application distribuée à base du paradigme orienté objet, en suite on passe à la conception et la réalisation d'un Modèle SMA primaire et en fin un Modèle SMA plus dynamique, extensible et adaptable à l'architecture de notre système d'information universitaire.

Dans ce chapitre nous présentons une description de l'existant dans le système d'information de l'université qui présente l'environnement d'intégration de cette première application (Scolarité en ligne), en particulier l'environnement numérique de travail (ENT) [61] et en suite une description détaillée de la problématique. En fin nous aboutissons l'architecture adoptée dans la distribution de l'application (Scolarité en ligne) à base du paradigme objet, plus sa modélisation et sa réalisation.

III.2 Analyse de la problématique et étude technique

III.2.1 Analyse de l'existant et problématique

Notre système d'information universitaire est distribué vue l'architecture réseaux de la présidence et les différents établissements universitaires. Plusieurs services de l'université disposent des applications statiques monopostes, et d'autres services utilisent encore les méthodes administratives classiques. Parmi ces services nous nous intéressons dans ce travail aux problèmes connus dans la procédure classique de la scolarité, en particuliers les demandes des attestations, des relevés de note, etc. L'automatisation de ce service nécessite une application distribuée, dynamique et extensible, du fait que le nombre des établissements de l'USMS augmente. Nous avons

pensé dans un premier temps à un modèle distribué à base du paradigme objet, que nous présentons dans ce qui suit. Cette application répond aux besoins de tous les établissements actuelles d'où la nécessité de l'intégrée dans l'ENT.

L'ENT est un espace numérique de travail conçu pour répondre aux besoins spécifiques des établissements d'enseignements. Ce portail est accessible en ligne depuis n'importe quel ordinateur connecté à Internet, aussi bien à l'intérieur qu'à l'extérieur de l'établissement et présente le point unique et sécurisé pour accéder à l'ensemble des outils et des services numériques de l'université. L'ENT participe à la structuration du paysage du numérique éducatif. L'intégration du projet ENT [62] au sein du système d'information de notre université permet à l'ensemble des utilisateurs d'avoir, grâce à une seule authentification, un accès en tout lieu et en tout point aux ressources personnelles et aux espaces de groupes auxquels ils appartiennent.

L'ENT USMS permet d'intégrer les différentes briques du système d'information, s'appuyant sur des interfaces normalisées entre les applications existantes et l'espace numérique de travail (interopérabilité applicative). L'application objective que nous présentons dans ce chapitre seras intégrées et accessible par L'ENT de l'université USMS.

La figure 10 présente la page d'accueil de L' ENT de USMS après authentification :

Figure 10: Page d'accueil de l'ENT

III.2.2 Description fonctionnelle de notre application

Après l'authentification de l'étudiant à l'ENT USMS, qui intègre cette application (scolarité en ligne), il peut choisir le service de scolarité pour faire une demande (Attestation de scolarité ou relevé de notes), ou envoyer un message au service s'il y a des problèmes. L'étudiant peut également consulter et suivre son historique des opérations effectuées. L'application offre à l'administrateur plusieurs possibilités qui lui permettent un contrôle total des différentes demandes effectuées de la part des étudiants. Ceci est garanti par un tableau de bord réalisé avec soin pour faciliter la gestion des différentes demandes. Plus particulièrement l'administrateur a la possibilité de répondre à une demande et modifier le statut d'une demande (mettre en cours, signée, disponible ou refusée). Selon le cas, l'administrateur peut supprimer ou archiver une demande.

L'application contrôle aussi le nombre de fois possibles d'envoi des demandes grâce à une base de données. La figure ci-dessous présente l'architecture du service de scolarité dans l'ENT.

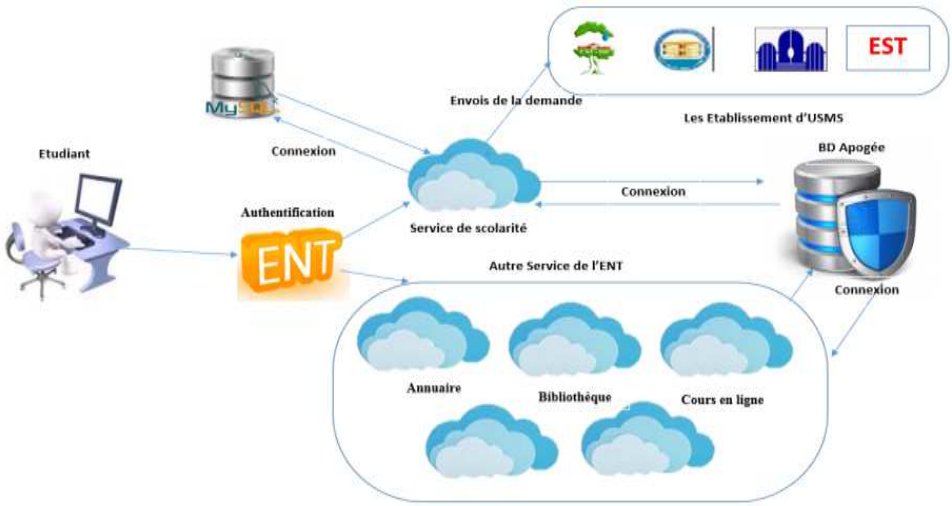


Figure 11 : L'architecture de service de scolarité dans l'ENT

III.2.3 Choix technique de l'architecture et des outils

Après avoir présenté une généralité sur l'existant et la problématique, nous présentons dans cette partie les différents outils et les technologies adoptés et utilisés pour la réalisation de notre application. Comme nous l'avons signalé auparavant, cette application est distribuée à base du paradigme orienté objet. Plusieurs méthodologies

et outils sont proposés dans le domaine de génie logiciel pour la réalisation des applications distribuées orientées objet. Nous allons faire un tour d'horizon, des différents choix techniques effectués, pour commencer la conception de l'application.
Comme dans tout projet informatique, cette application est créée, pour répondre aux besoins et aux spécifications fonctionnelles, la réalisation de ces derniers devrait résoudre notre problématique. Pour créer une application qui satisfait les spécifications fonctionnelles et techniques, une application aussi bien performante avec des qualités de maintenance et de sécurité et surtout avec un coût de temps acceptable, il faut tout d'abord bâtir l'application sur des architectures existantes pour bénéficier de l'expérience des autres et des recherches faites dans ce domaine.
Dans le choix de l'architecture et des outils, les critères suivants sont pris en compte :

- L'application doit être basée sur une architecture distribuée performante ;
- Les outils de développements devrons être open source ;
- L'application doit être consultée avec un navigateur web ;

Nous avons donc adopté dans notre projet, l'architecture distribuée 3-tiers à base de la Plateforme J2EE (Java Enterprise Edition) et le pattern MVC, ce qui permet de décomposer l'application en trois couches (niveaux), comportant un serveur web J2EE, un serveur métier J2EE et un serveur de base de données pour avoir un découpage entre les différentes parties de l'application.

JEE (Java Enterprise Edition) facilite le développement d'application d'entreprise en fournissant un environnement d'exécution et des composants sous forme d'API (application programming interface).

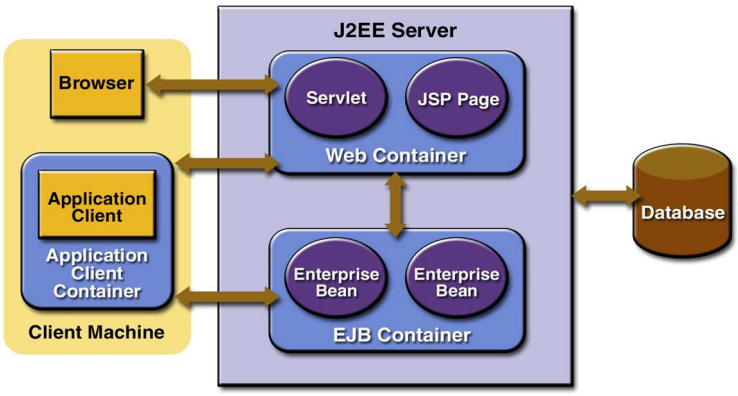


Figure 12: Architecture d'une application web J2EE

Le modèle MVC (Modèle, Vue, Contrôleur), est un design pattern permet de découper littéralement l'application en couches distinctes, pour l'organisation du code :

- Un contrôleur est implémenté sous forme de Servlet Java,
- Le modèle consiste en l'implémentation du logique métier de l'application ;
- La vue est implémentée via une JSP (Java Server Pages).

Servlets implémentent la couche contrôleur du modèle MVC, pour traiter les requêtes issues d'un navigateur web client et générer une sortie dynamique qui est retournée en HTML au client.

Les JSP, implémentent la couche Vue du MVC et présentent les pages web.

Apache Tomcat : le serveur d'application qui gère les servlets

NetBeans est l'environnement de développement Intégré (EDI), que nous avons adopté pour réaliser notre application à base de JEE.

Oracle SQL Developer représente l'environnement qu'on a utilisé pour se connecte à la base de données Apogée pour exécuter les requêtes et faire de test car apogée est une base de données oracle. Il nous a permis d'afficher les descriptions des tables et des colonnes des tables ce qui aide à comprendre les relations entre les tables.

Power AMC : nous avons utilisé ce logiciel dans la modélisation UML et d'autres.

Bases de données : notre application utilise deux bases des données :

- une base de données Oracle (Apogée) qui contient les données des étudiants, on a besoin de se connecter à cette base pour extraire les informations des étudiants (informations personnelles, notes des semestres et des modules, année d'inscription, établissement dans laquelle il est inscrit etc.…).
- une base de données MySQL crée et consacrée aux données nécessaires pour stocker les demandes et les comptes des administrateurs de notre application.

III.3 Modélisation de cette application objective

III.3.1 UML pour la modélisation

Etant donné que nous avons achevé l'étude technique du projet, et nous avons décrit la solution proposée selon une spécification des besoins dans la phase préalable, nous dédions cette partie à la modélisation. Plusieurs langages de modélisation sont présents dans le domaine de génie logiciel pour la modélisation des différents paradigmes, nous utiliserons dans la modélisation de cette application le langage de modélisation UML, qui présente un langage de modélisation très puissant pour la programmation orienté objet.

Pour la modélisation des besoins, nous utilisons les diagrammes UML suivants : Diagramme de cas d'utilisation, diagramme de séquence, diagramme de collaboration et diagramme d'activité.

III.3.2 Diagrammes des cas d'utilisation

Les figures suivantes résument la première étape de la phase de la modélisation de notre projet, elles présentent des diagrammes des cas.

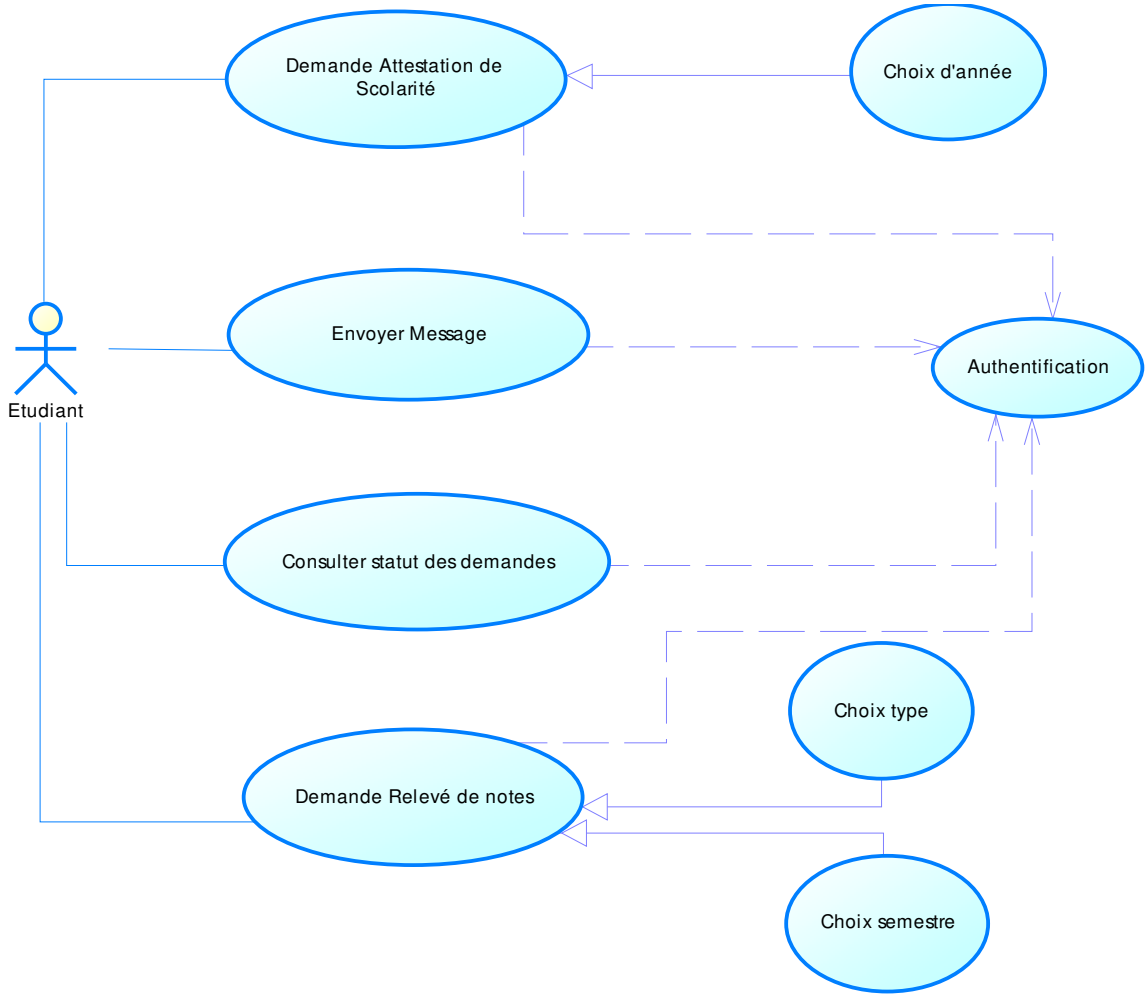


Figure 13: Diagramme des cas d'utilisation Etudiant

Ce diagramme des cas d'utilisation a pour objectif de déterminer ce que chaque utilisateur attend du système. La détermination du besoin est basée sur la représentation de l'interaction entre l'acteur et le système.

L'acteur dans ce cas c'est l'étudiant qui interagit avec l'application en choisissant le type de demande et en entrant ses informations personnelles nécessaires.

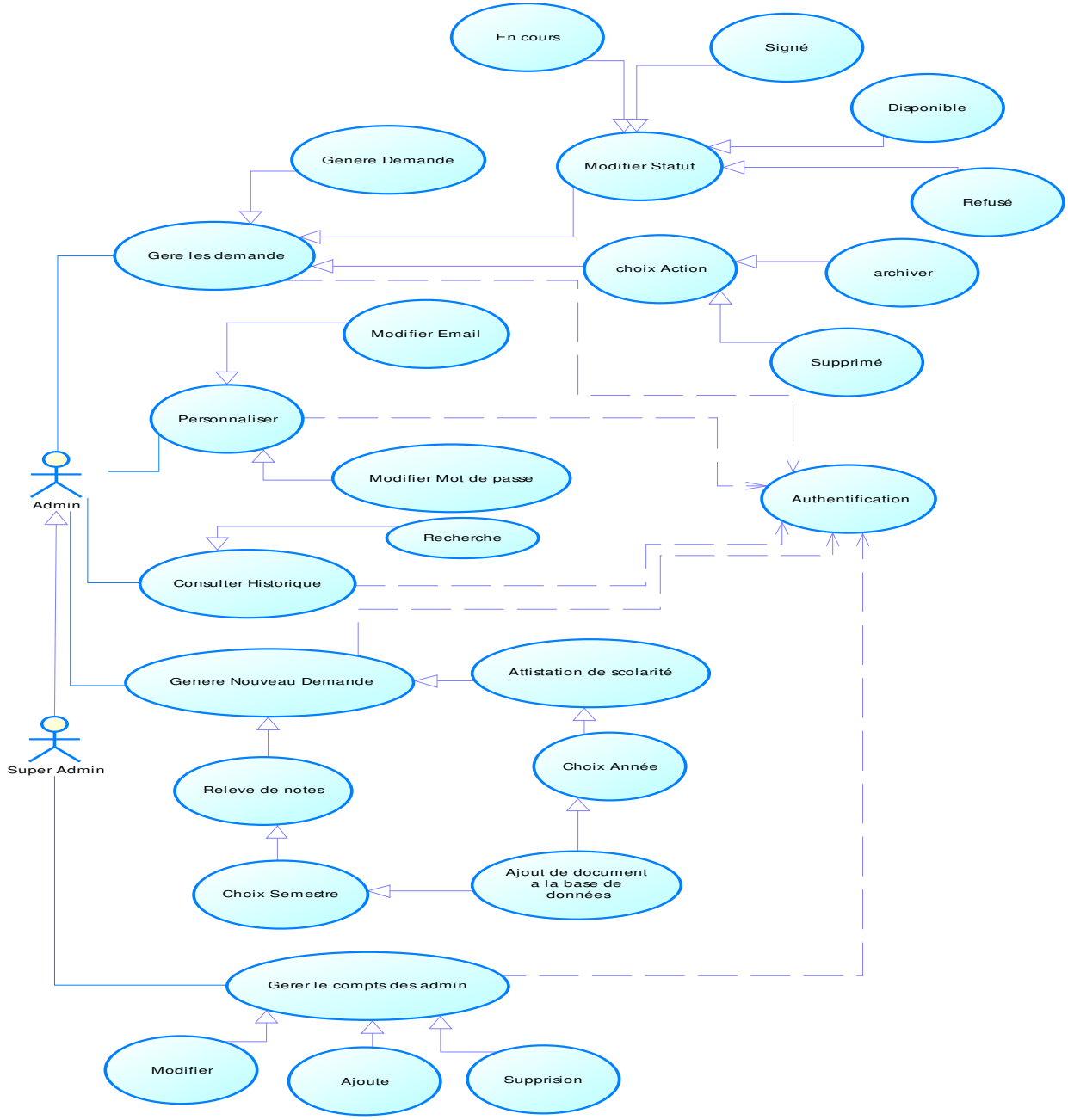


Figure 14: Diagramme de cas d'utilisation Admin et Super Admin

Ce diagramme de cas d'utilisation a pour objectif de présenter l'interaction entre les administrateurs (agent de scolarité) qui traitent les demandes des étudiants avec les entités de l'application, ainsi le cas pour le Super Administrateur, qui représente le responsable technique sur l'application.

III.3.3 Model des tables Apogée utilisées

APOGEE est une application développée par l'Agence de mutualisation des universités et des établissements(AMUE) depuis 1995. Elle est basée sur une base de données

oracle et destiné à la gestion des inscriptions et des dossiers des étudiants dans les universités.

La figure ci-dessous présente un PowerMCD modèle de la table « INDIVIDU » et d'autres tables de la base apogée, utilisée pour la création des attestations scolaires.

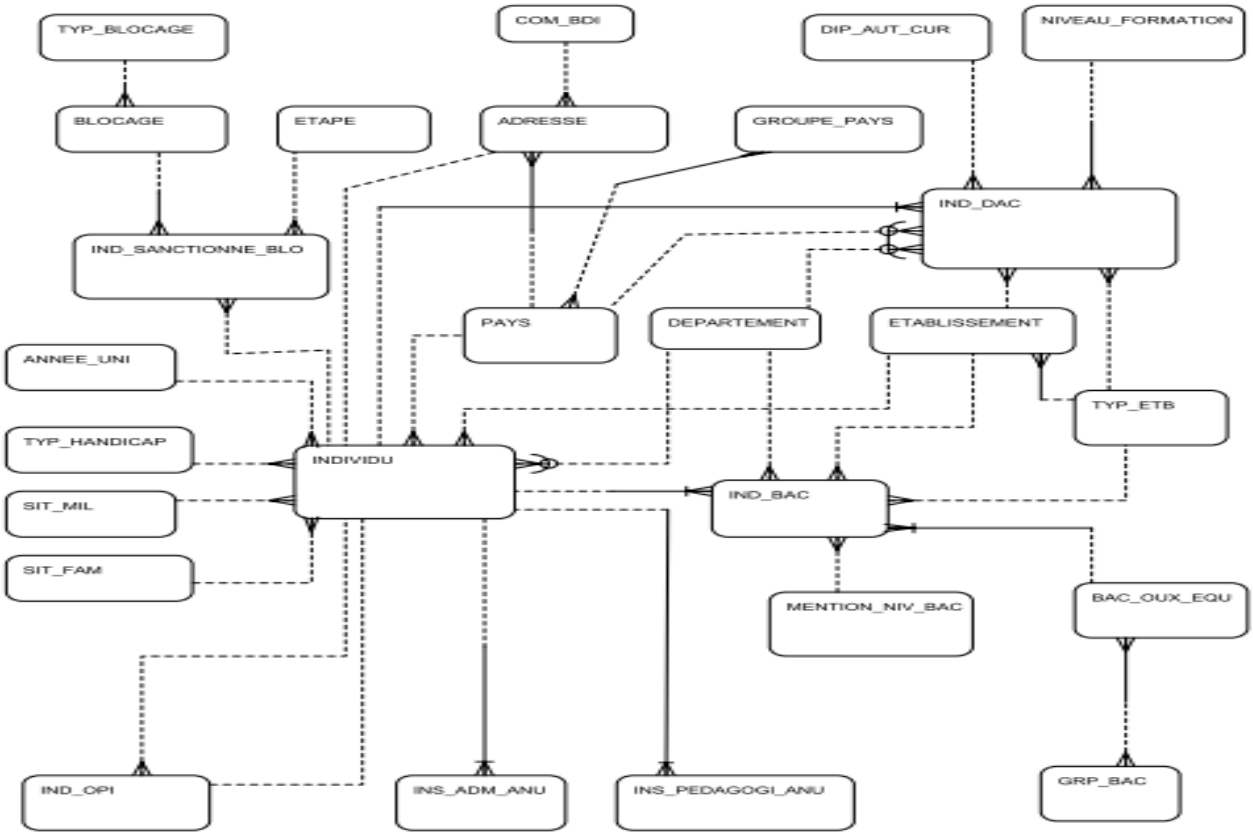


Figure 15: MCD - INDIVIDU / ETUDIANT de la base de données Apogée

Nom de table	Description
ADRESSE	Tables des adresses d'un individu
ANNEE_UNI	Année universitaire : période durant laquelle un établissement dispense des enseignements
COMPOSANTE	Composantes (facultés, FST, FP...) : entité ayant une responsabilité de type pédagogique
DIPLOME	Table des titres délivrés par l'établissement après validation d'une ou plusieurs étapes
ETABLISSEMENT	Etablissements Marocaine dispensant des enseignements secondaires ou supérieurs
ETAPE	Table des étapes : fractionnement dans le temps de la préparation d'un diplôme, d'un concours
INDIVIDU	Table des individus
INS_ADM_ANU	Dossiers d'inscription administrative annuelle
PAYS	Table des pays

Tableau 3: Description des tables utilisée pour la création des attestations de scolarité

La figure ci-dessous présente un Power MCD modèle des tables de la base apogée, utilisées dans l'extraction des relevés de notes.

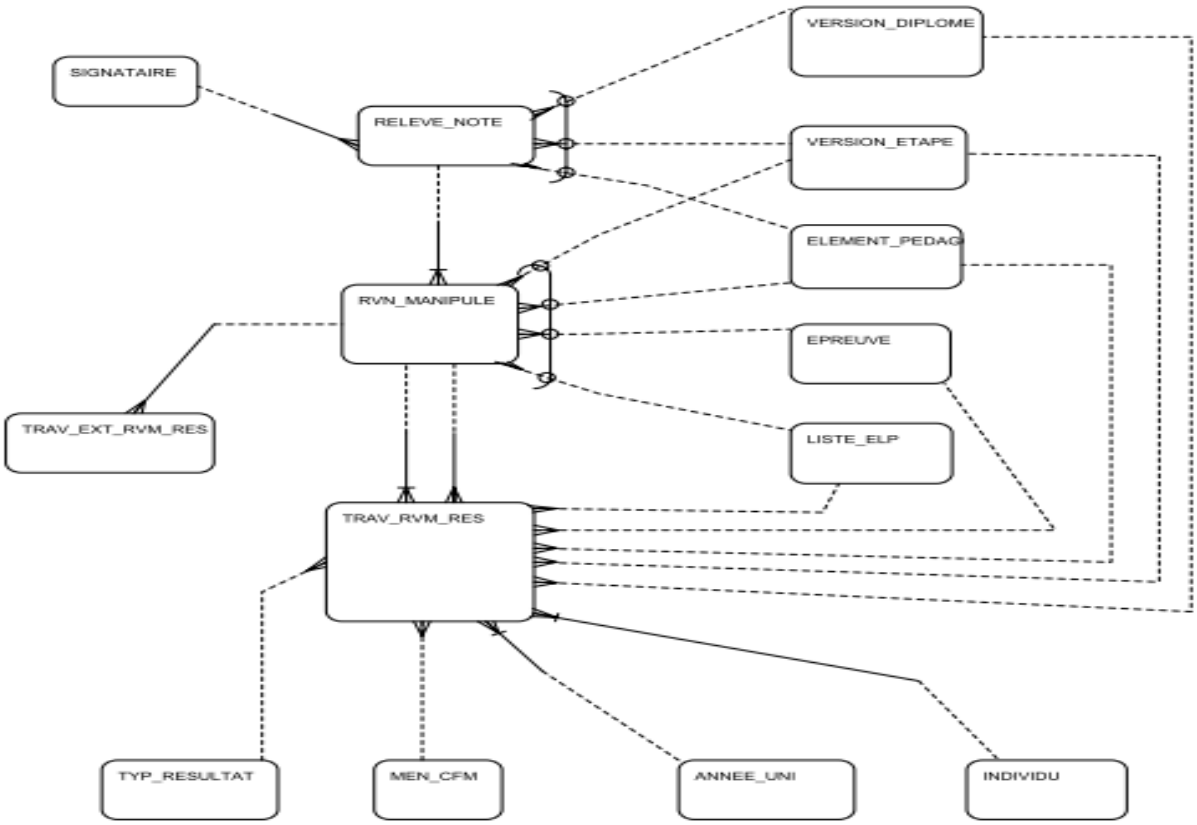


Figure 16: MCD - COLLECTE ET DIFFUSION DES RESULTATS : RELEVE DE NOTES

Nom de table	Description
ANNEE_UNI	Année universitaire : période durant laquelle un établissement dispense des enseignements
COMPOSANTE	Composantes (facultés, FST, FP...) : entité ayant une responsabilité de type pédagogique
DIPLOME	Table des titres délivrés par l'établissement après validation d'une ou plusieurs étapes
ETABLISSEMENT	Etablissements Marocain dispensant des enseignements secondaires ou supérieurs
ETAPE	Table des étapes : fractionnement dans le temps de la préparation d'un diplôme, d'un concours
INDIVIDU	Table des individus
INS_ADM_ANU	Dossiers d'inscription administrative annuelle
PAYS	Table des pays
ELEMENT_PEDAGOGI	Éléments pédagogiques de description des enseignements

RELEVE_NOTE	Relevé de notes
RESULTAT_ELP	Résultats des étudiants aux éléments pédagogiques
RVN_MANIPULE	Table de manipulation de relevé de notes
MENTION	Mention / Configuration de mention
TYP_RESULTAT	Types de résultats
TRAV_RVM_RES	Table de travail relevé note / résultats
SIGNATAIRE	Signataires des attestations de réussite

Tableau 4 : Description des tables utilisées pour la création des attestations de scolarité

III.3.4 Diagramme de classes

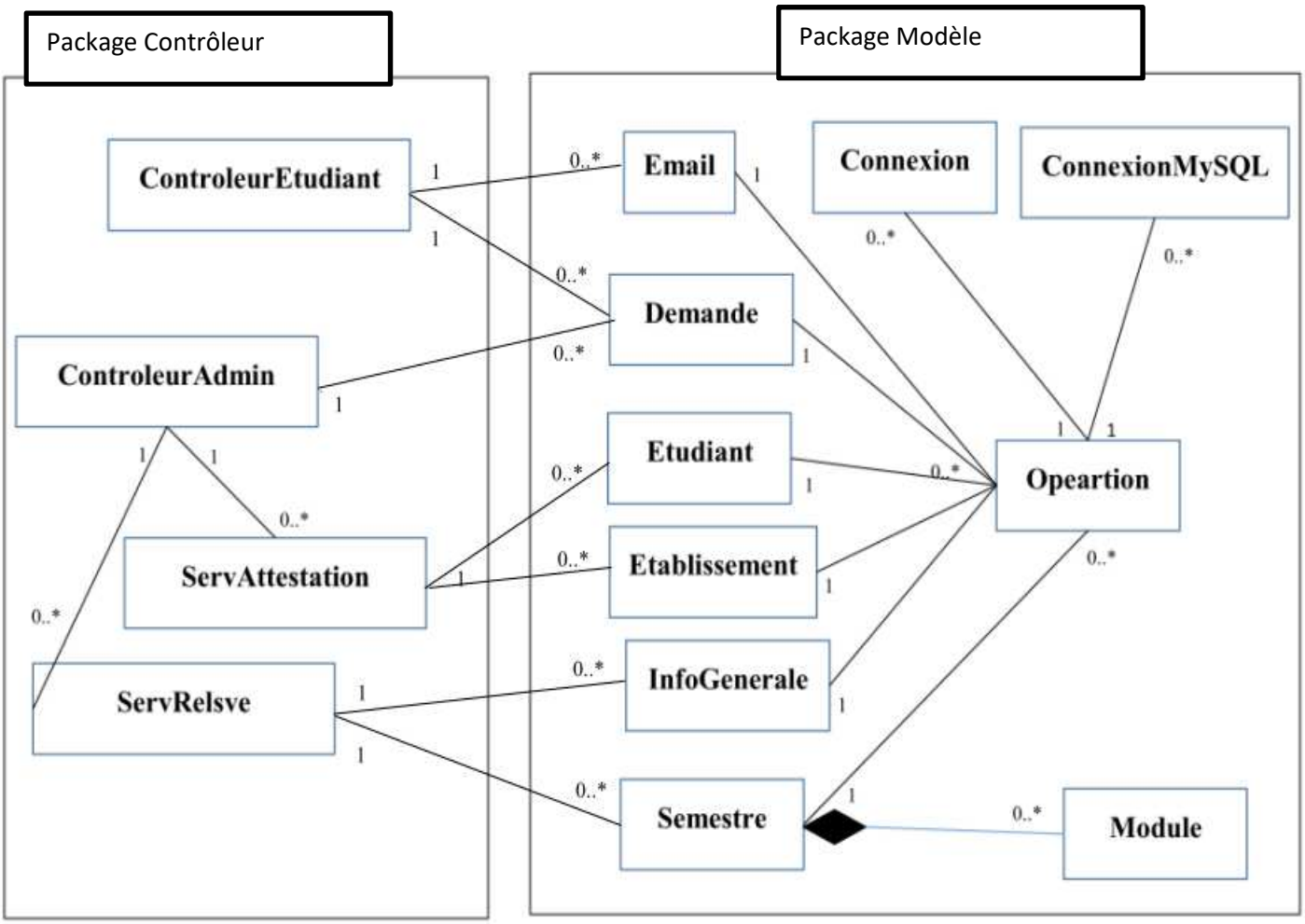


Figure 17: Diagramme de classes de la package Modèle

III.3.5 Diagramme de séquences

Pour décrire les scénarios de chaque cas d'utilisation déjà élaborés et de diagramme de classe, et à fin de mieux représenter les opérations en interactions avec les objets, une

interaction séquence dans le temps, nous avons utilisé des diagrammes de séquences. Vue les grandes tailles de quelques diagrammes de séquences, nous présentons dans la suite que les diagrammes, les plus lisibles.

Diagramme de séquence d'envoyer message

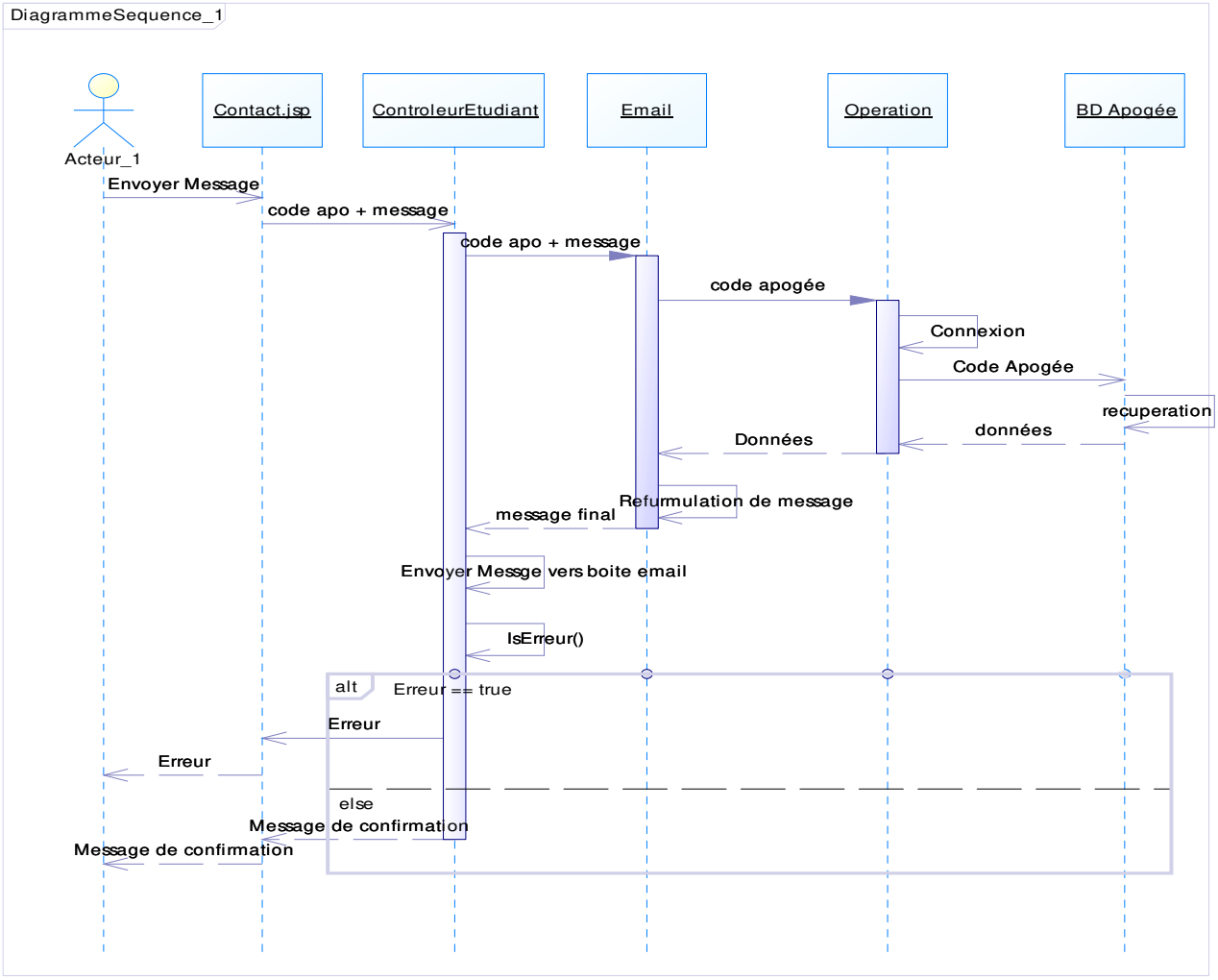


Figure 18: Diagramme de séquence d'envoyer un message

Diagramme de séquence de création de relevé de notes par administrateur

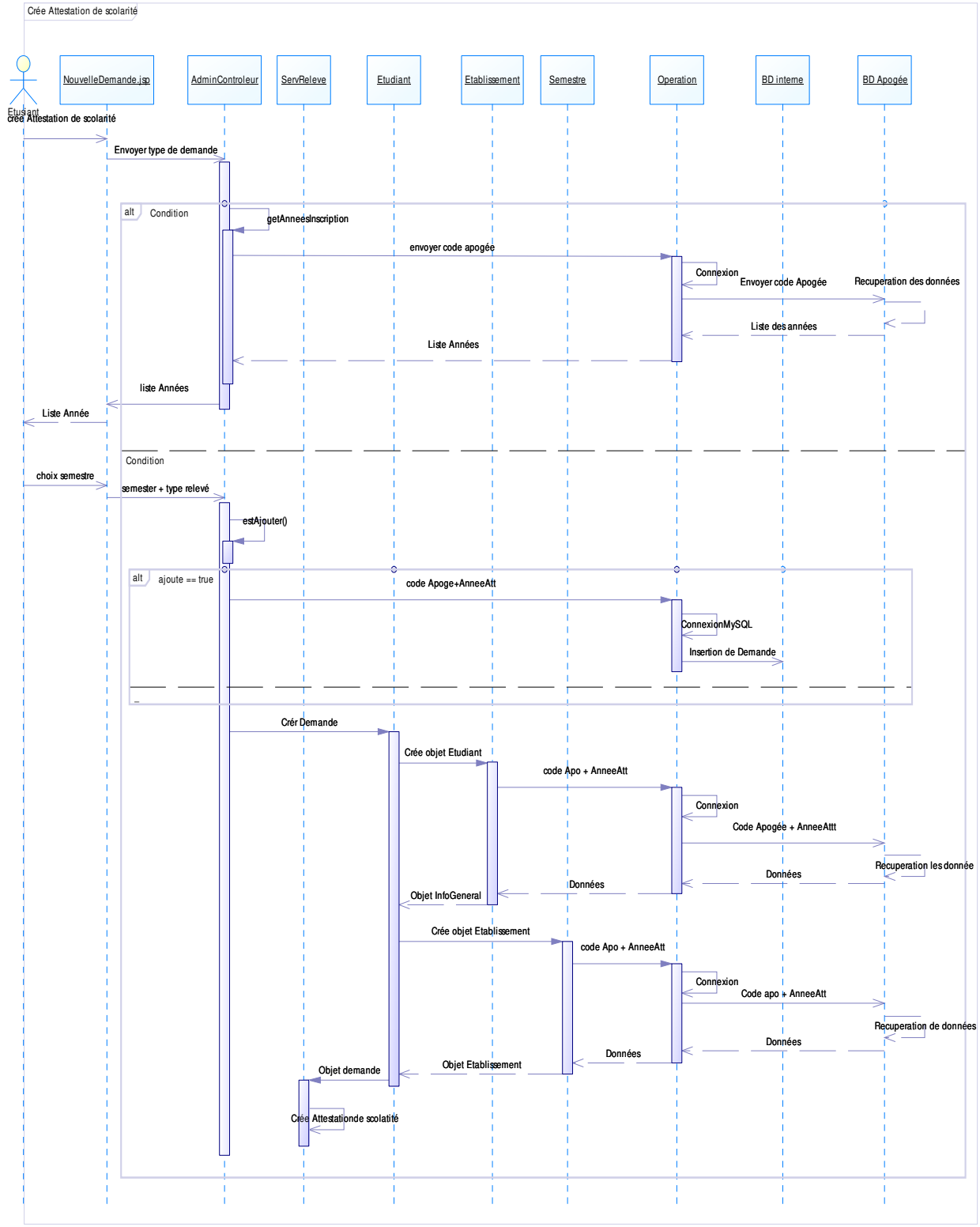


Figure 19: Diagramme de séquence de création attestation de scolarité par administrateur

III.3.6 Conception de la base de données interne

Pour contrôler le nombre de demandes effectuées par l'étudiant durant l'année, la base de données Apogée ne permet pas de résoudre ce problème c'est pourquoi on a ajouté une deuxième base de données SQL.

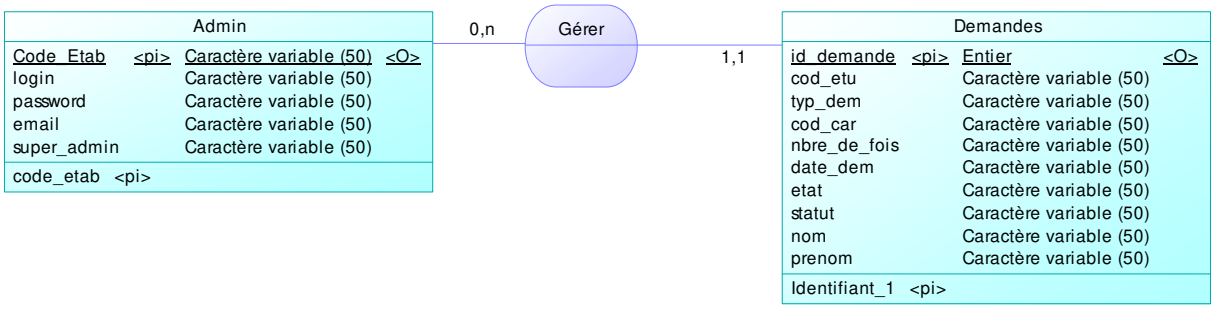


Figure 20: Modèle conceptuel des données (MCD) de la base de données interne

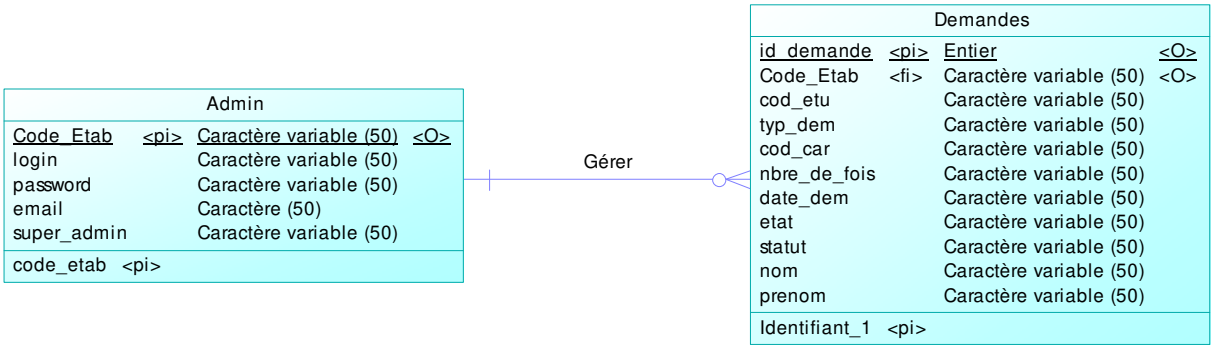


Figure 21: Modèle logique des données (MLD) de la base de données interne

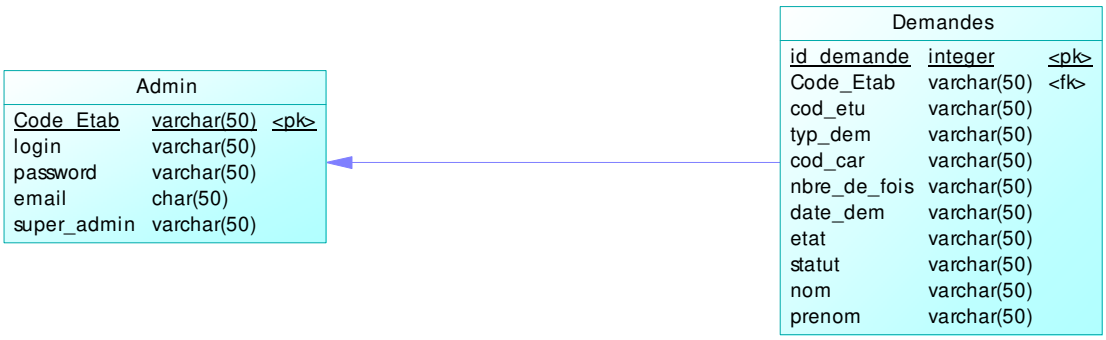


Figure 22: Modèle Physique des Données (MPD) de la base de données interne

III.4 Phase expérimentale et résultats

III.4.1 La base de données APOGEE

La base apogée est une base oracle, ainsi nous avons utilisé Oracle Développer pour accéder à cette base, sélectionner et éditer les tables dont on a besoin de colonnes et description. La base apogée contient (371 tables).

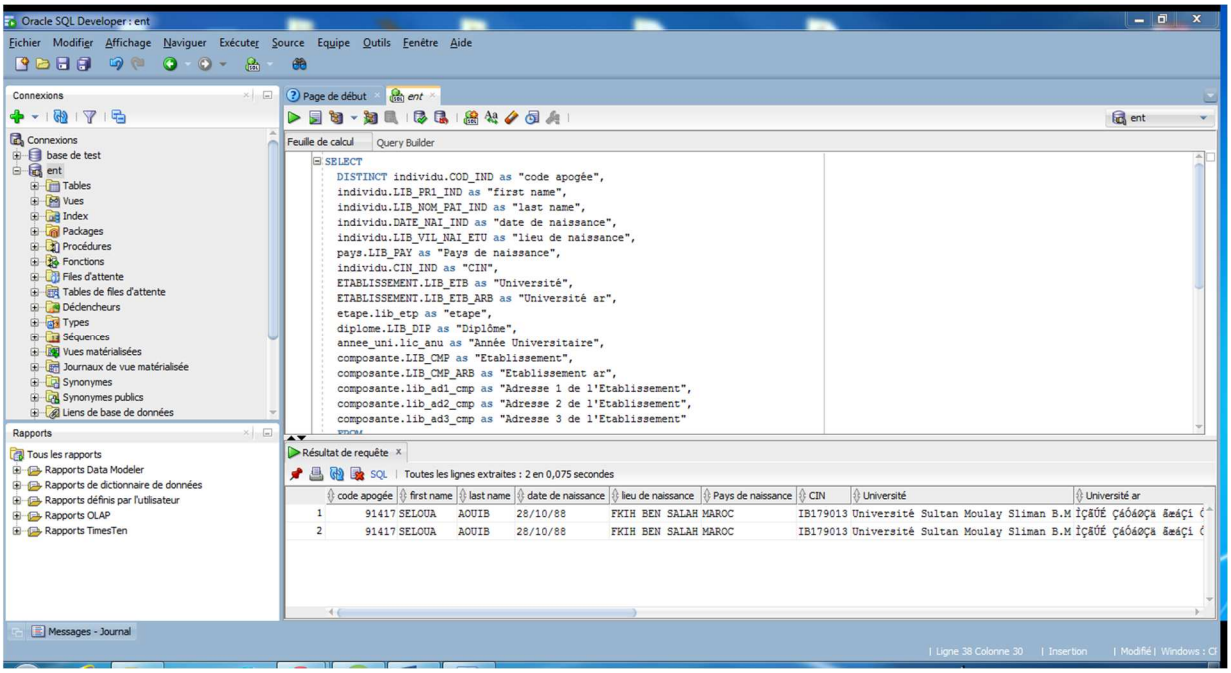


Figure 23: Informations sur les colonnes de table individu fournit par « Oracle Developer »

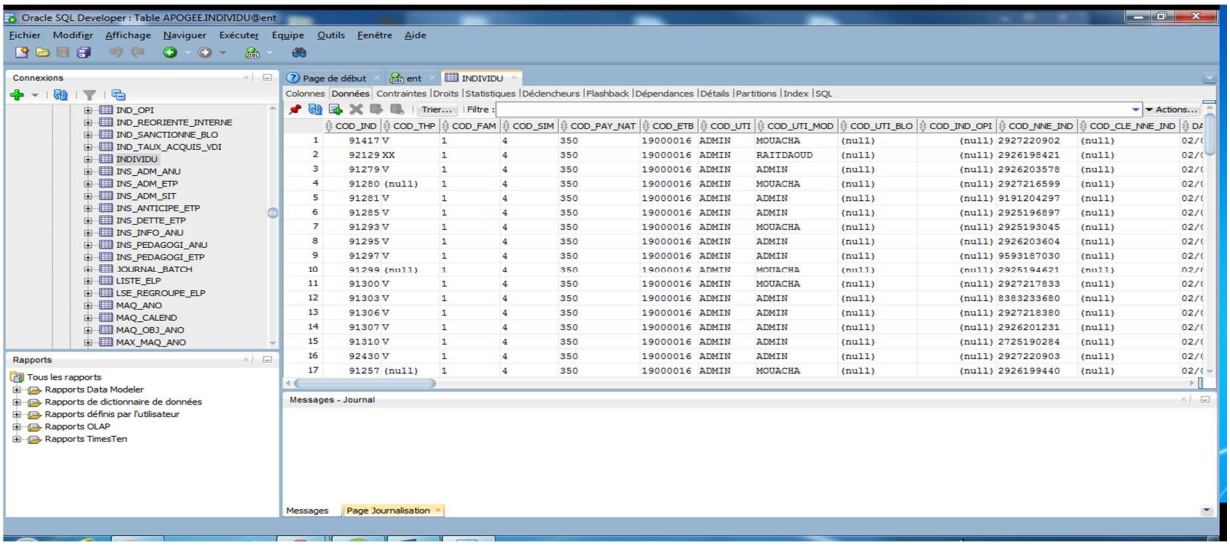


Figure 24: Exemple d'exécution de requête sur « Oracle Developer »

III.4.2 Résultats et quelques pages web de notre application

Nous présentons quelques étapes de la procédure de fonctionnement de cette application. L'interface de cette application offre à l'étudiant et aux administrateurs le choix de plusieurs fonctionnalités en navigant celons différents onglets, nous citons les plus importantes.

- ✓ Une fois l'étudiant accède à cette page, il choisit soit de demander une attestation de scolarité ou une demande de relevé de Notes

Figure 25: Page des demandes

- ✓ Un message de confirmation s'affiche si la demande passe correctement sans aucun problème.

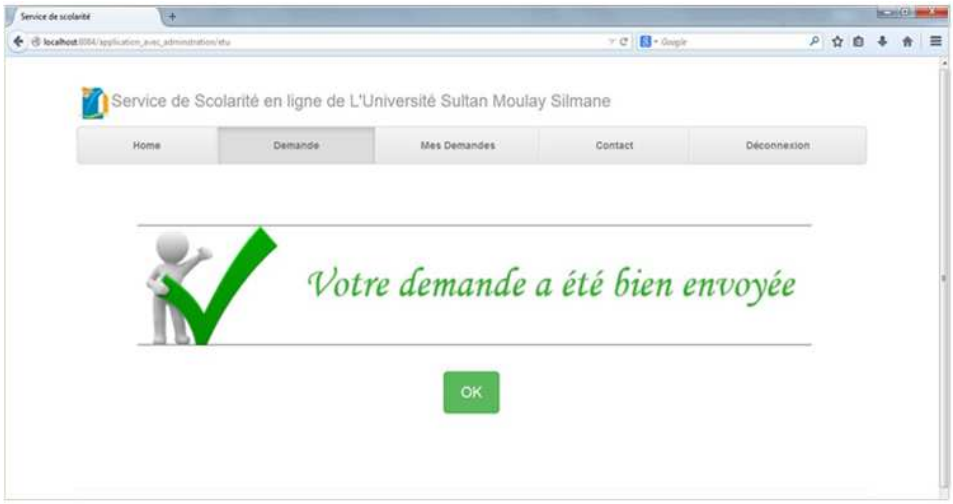


Figure 26: Page de confirmation

✓ Une fois l'étudiant a atteint le nombre maximal de demandes offertes (fixé en trois par année), un message d'erreur est affiché.

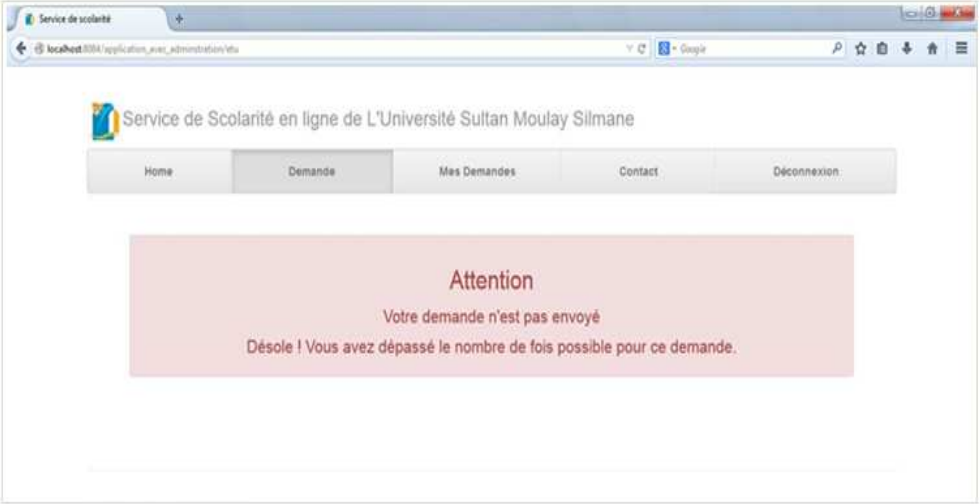


Figure 27: Affichage de message d'erreur

✓ Lorsque l'utilisateur a effectué une demande, et que le statut de cette demande est en cours, si l'étudiant fait la même demande le message suivant est affichée.

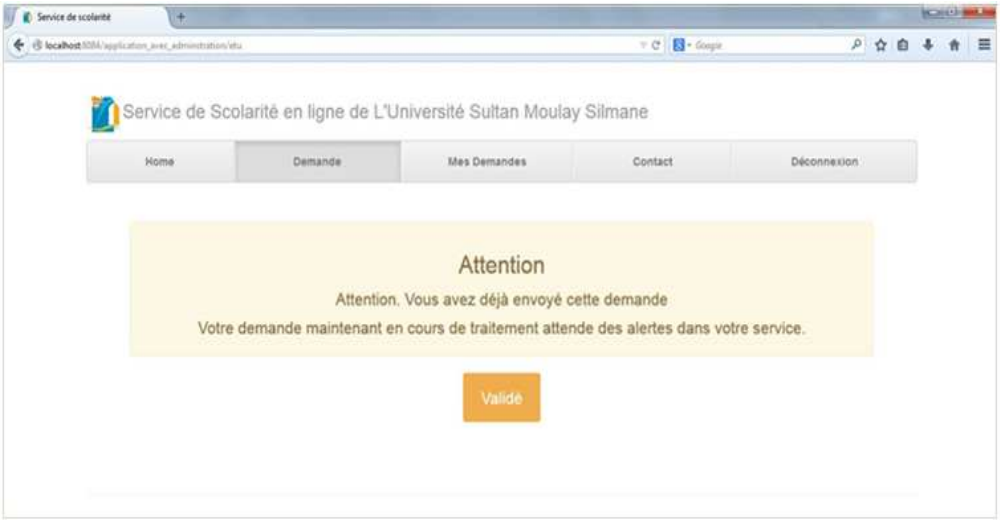


Figure 28: Statut de la demande

Pour communiquer avec la scolarité, cette application offre à l'étudiant la possibilité d'envoyer un message au service grâce à la partie contact du service.

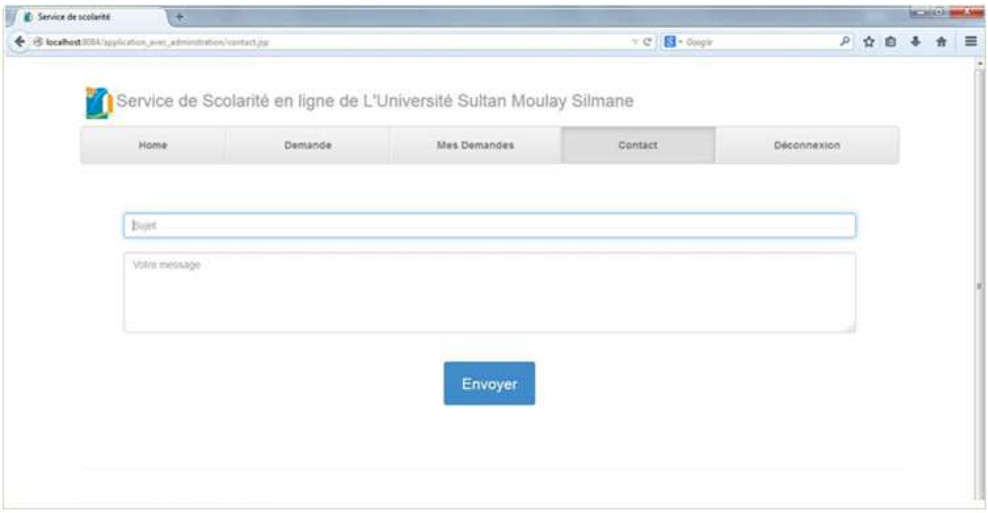


Figure 29: Page contact

Une fois le message envoyé, cette zone de contact permet aussi d'envoyer avec le message toutes les informations concernant l'étudiant envoyant le message. Et l'interface suivant est affiché pour indiquer que le message a été bien envoyé.

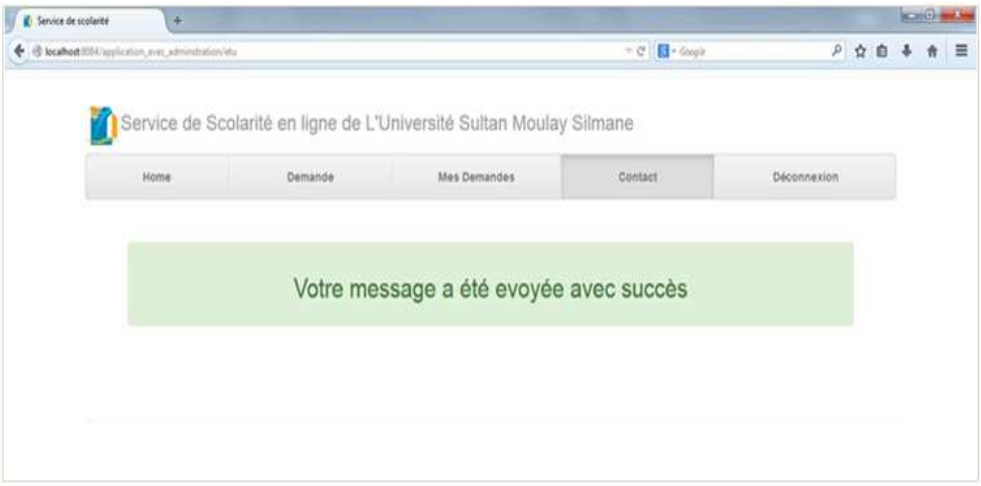


Figure 30: Message de confirmation

L'accès à la partie administration se réalise par un login et mot de passe présenté par la figure 31.

Figure 31: Page login

Vue le nombre important des interfaces que représentes les actions dans cette application, nous présentons que les importantes et nous résumons les actions.

- ✓ L'administrateur est connectée l'application propose plusieurs actions selon le choix :
- ✓ généré, automatiquement les demandes déposées dans l'application ;
- ✓ crée un document (attestation de scolarité et/ou relevé de notes) ;
- ✓ consulter l'historique de toutes les demandes ;
- ✓ recherche avec plusieurs options ;
- ✓ Impression des documents générés ;

Résultats : attestation de Scolarité et relevé de note.


Université Sultan Moulay Slimane
Faculté Polydisciplinaire
Béni Mellal

جامعة السلطان مولاي سليمان
الكلية المتعددة التخصصات
بني ملال

ATTESTATION DE SCOLARITE

Le Doyen de Faculté Polydisciplinaire Béni Mellal atteste que etudiant :

Monsieur [redacted]

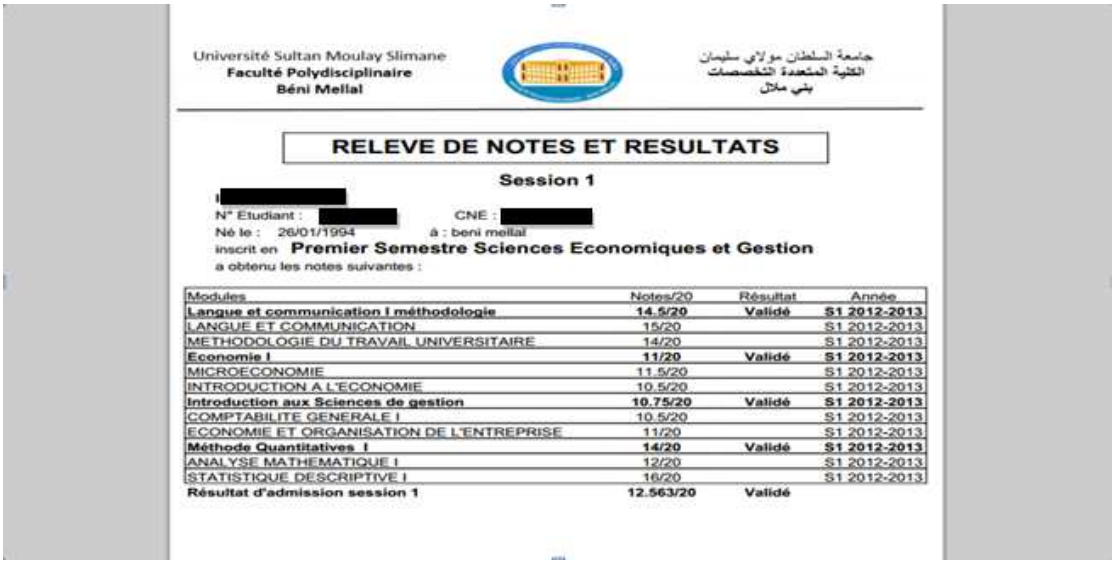
Numéro de la carte d'identité nationale : IC 84909

Code national de l'étudiante : 1210195180

née le 13/01/1993 à Afourer (MAROC)

était régulièrement inscrite à Faculté Polydisciplinaire Béni Mellal Béni Mellal pour l'année universitaire 2012/2013.

Figure 32: Exemple d'attestation de scolarité



Université Sultan Moulay Slimane
Faculté Polydisciplinaire
Béni Mellal

جامعة السلطان مولاي سليمان
الكلية المتعددة التخصصات
بني ملال

RELEVE DE NOTES ET RESULTATS

Session 1

[redacted]
N° Etudiant : [redacted] CNE : [redacted]
Né le : 26/01/1994 à : beni mellal
inscrit en **Premier Semestre Sciences Economiques et Gestion**
a obtenu les notes suivantes :

Modules	Notes/20	Résultat	Année
Langue et communication I méthodologie	**14.5/20**	**Validé**	**S1 2012-2013**
LANGUE ET COMMUNICATION	15/20		S1 2012-2013
METHODOLOGIE DU TRAVAIL UNIVERSITAIRE	14/20		S1 2012-2013
Economie I	**11/20**	**Validé**	**S1 2012-2013**
MICROECONOMIE	11.5/20		S1 2012-2013
INTRODUCTION A L'ECONOMIE	10.5/20		S1 2012-2013
Introduction aux Sciences de gestion	**10.75/20**	**Validé**	**S1 2012-2013**
COMPTABILITE GENERALE I	10.5/20		S1 2012-2013
ECONOMIE ET ORGANISATION DE L'ENTREPRISE	11/20		S1 2012-2013
Méthode Quantitatives I	**14/20**	**Validé**	**S1 2012-2013**
ANALYSE MATHEMATIQUE I	12/20		S1 2012-2013
STATISTIQUE DESCRIPTIVE I	16/20		S1 2012-2013
Résultat d'admission session 1	**12.563/20**	**Validé**	

Figure 33: Exemple de relevé de notes détaillé

Remarque : Comme caractéristique de cette application, elle peut s'adapter à n'importe quel écran soit d'un ordinateur ou bien une tablette, mobile, etc.

Figure 34: Vue de l'application sur un mobile (320x480)

Figure 35: Vue de l'application sur un mobile (800x600)

III.5 Conclusion

Dans ce chapitre, nous avons présenté la modélisation et la réalisation de l'application en justifiant nos choix technologiques, en représentant quelques interfaces graphiques que nous avons jugé les plus importantes et en décrivant brièvement comment nous avons planifié cette application. L'objectif de cette application est atteint. Nous avons mené ce travail en parallèle avec nos recherches dans ce domaine qui est très important du fait qu'on a tendance aujourd'hui à dépasser le monde des logiciels statiques et monoposte (basés dans une seule machine isolée) vers les logiciels distribuées et dynamique.

La distribution de cette application en 3 couches (client, serveurs d'application, serveurs des données) nous a permis d'éviter le mode monoposte qui demande l'installation et la configuration pour chaque utilisateur. Mais cette distribution reste insuffisante, du fait qu'elle est non adaptable aux nombre d'établissements universitaire qui évolue, dans un cas d'évolution, il faut changer tout le code qui est compact dans le serveur d'application, d' où la nécessité d'une autre discipline de distribution plus flexible et adaptable. Nous avons adopté dans la suite l'intelligence artificielle distribuée (IAD) à base des systèmes multi-agents, qui ont un haut niveau d'abstraction, de dynamisme et de flexibilité (vue chapitre II).

Chapitre IV : Modèle SMA1 pour la gestion de scolarité.

IV.1 Introduction

Ce chapitre est consacré à l'exploitation des études théoriques sur le paradigme agent, aboutit dans les premiers chapitres de cette thèse dans le but de réaliser un modèle de scolarité à base des SMA, une application à base des agents, ce modèle SMA1 a le même objectif que celui de l'application présentée dans le chapitre précédent, qui est la réalisation d'un service de scolarité, mais cette fois à base des agents. L'intérêt que ce modèle suscité est lié à la capacité des SMA d'aborder les problèmes complexes d'une manière distribuée et de proposer des solutions réactives et robustes.

La première partie de ce chapitre est consacrée à la présentation de l'architecture, les différents outils et les technologies adoptées et utilisées pour la réalisation de cet objectif SMA1.

La deuxième partie présente l'aspect d'analyse, de la modalisation et d'implémentation de ce modèle. Plus l'ensemble des fonctionnalités avec une description des différentes interfaces de cette application.

Pour répondre aux exigences, le système multi-agent que nous proposons est basé sur des entités appelées agents communiquant les uns avec les autres pour coordonner leur comportement intelligent dont le but d'atteindre l'ensemble des objectifs du système.

IV.2 Choix de la plateforme et outils de modélisation

Nous avons présenté dans le deuxième chapitre, plusieurs plateformes d'agents, qui facilitent le développement et la gestion des systèmes multi-agents. Le choix de la plateforme d'agent dépend fortement de l'objectif de l'application en question. Dans le cadre de notre travail, la plateforme doit répondre aux exigences suivantes :

- Le système multi-agent objectif est distribué, la plateforme nécessaire doit supporter le développement des modèles SMA distribués ;

- Création des agents interactifs et communicatifs cordonnant leur comportement intelligent ;
- La plateforme doit utiliser un Langage de programmation respectant les normes FIPA ;
- Une plate-forme riche en termes de documentation est plus facile à utiliser.

En se basant sur la comparaison de plusieurs plateformes : MadKit, JADE, GAMA et JADEX, faites précédemment, et les critères cités ci-dessus. Notre choix est axé sur la plateforme JADE qui répond à nos exigences.

IV.3 Plateforme multi-agents JADE

La plateforme JADE est utilisée pour l'implémentation et la gestion des agents.
JADE est complètement développée en JAVA, suit les spécifications émises par l'organisme FIPA (Fondation for Intelligent Physical Agent) et une organisation de chercheurs industriels et académiques ayant proposé de nombreux standards en lien avec les agents.
JADE est une plateforme qui facilite le développement des systèmes multi agents. Nous présentons dans ce qui suit une description des principales spécificités de JADE.

Cette plateforme contient :

- Un **Runtime Environment** : l'environnement ou les agents peuvent vivre, cet environnement d'exécution doit être actif pour pouvoir lancer les agents ;
- **Une librairie de classes** : que les développeurs utilisent pour écrire leurs programmes correspondants aux agents ;
- **Une suite d'outils graphiques :** qui facilitent la gestion et la supervision de la plateforme des agents.

JADE fournit aussi des classes qui implémentent JESS pour la définition du comportement des agents **:** JESS « outil de raisonnement à base de règles » est le moteur qui exécute tout le raisonnement nécessaire.

Chaque instance du JADE est appelée container « conteneur », et peut contenir plusieurs agents. Un ensemble de conteneurs constituent une plateforme. Chaque plateforme doit contenir un conteneur spécial appelé main-container « conteneur principal » et tous les autres conteneurs s'enregistrent auprès de celui-ci dès leurs lancement.

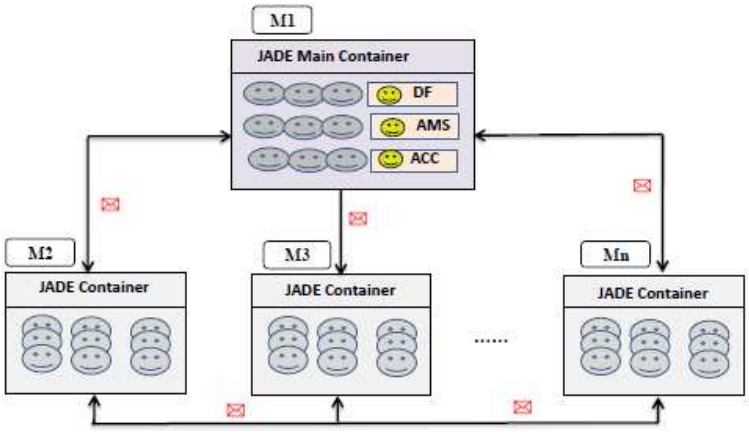


Figure 36: Architecture de la plateforme Jade

Un main-container se distingue des autres conteneurs car il contient toujours deux agents spéciaux appelés AMS et DF qui se lancent automatiquement au lancement du main-container. Ainsi on a un conteneur par machine et lorsque l'on lance un conteneur :

- soit il est seul, il devient alors le conteneur principal « main container ».
- soit on lui indique l'adresse du conteneur principal.

Les agents spéciaux de la plateforme jade sont :

- **AMS « Agent Management System »** : un Service de Pages Blanches qui référence automatiquement chaque agent en lui donnant un nom dès son entrée dans le système. Il permet aussi la communication interne entre les agents d'une plateforme ;
- **DF « Directory Facilitator »** : Service de Pages Jaunes : référence à leur demande les agents suivant leur(s) service(s) ;

- **RMA « Remote Monitoring Agent »** : cet agent d'interface est exclusivement réservé à la gestion de l'application JADE en cours d'exécution. Il permet d'afficher un certain nombre d'interfaces interrogeables permettant de renseigner l'utilisateur humain sur le déroulement de la session JADE. Ainsi, en interrogeant le RMA, on peut notamment obtenir des informations graphiques provenant des deux agents précédemment cités ;
- Agent Snifer : pour la surveillance des communications en s'assurant du bon déroulement des protocoles des communications ;
- **ACC« Agent Communication Channel » :** qui gère la communication entre la plateforme jade et d'autre plateforme.

Les caractéristiques des agents d'une plateforme JADE déployée sur une ou plusieurs machines :

- ✓ La plateforme héberge un ensemble d'agents, identifiés de manière unique, pouvant communiquer de manière bidirectionnelle avec les autres agents ;
- ✓ Chaque agent s'exécute dans un conteneur (container) qui lui fournit son environnement d'exécution ; il peut migrer à l'intérieur de la plateforme ;
- ✓ Les agents sous JADE sont menus d'une autonomie dans le sens de l'indépendance de fonctionnement. En pratique, cela se caractérise par leur capacité à effectuer un ensemble de comportements qui leur sont propres, sans la nécessité d'une interaction externe. Ces comportements sont appelées « Behavior ». JADE propose également des classes de bases pour des types de comportements spécifiques ;

La figure ci-dessous présente le processus de la communication des messages entre les agents.

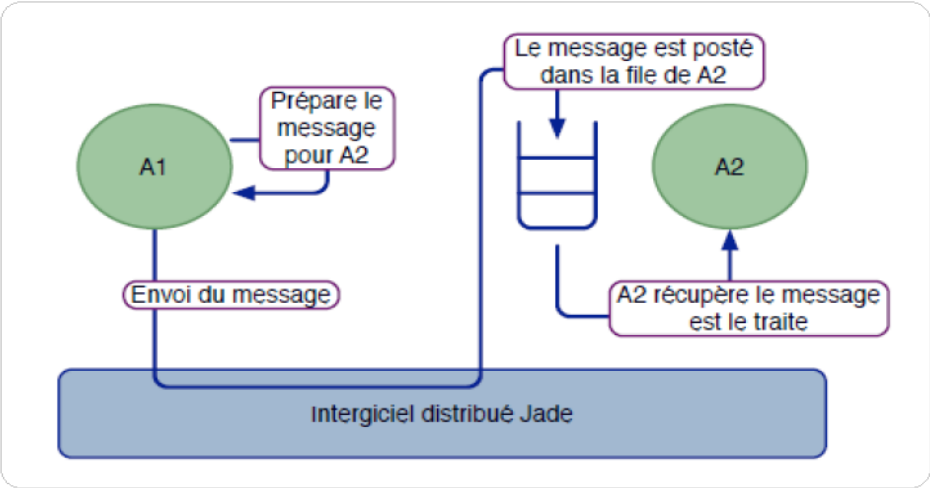


Figure 37: Paradigme du passage d'un message asynchrone

IV.4 AUML pour la modélisation

Comme nous l'avons déjà vue dans le chapitre 3 l'AUML est basé sur la méthode UML et du fait que les agents ont des activités autonomes et des buts par rapport à l'objet, C'est cette différence qui entraine l'insuffisance d'UML pour modéliser les agents et les systèmes multi-agents. Aussi AUML remplace la notion de méthode par celle de service.

Les principales extensions de L' AUML sont :

- Diagramme de classes d'agent qui est une reformulation du diagramme de classes d'objets ;
- Diagramme de séquence qui permet une meilleure modélisation des interactions entre les agents ;
- Diagramme de collaboration qui complémente le diagramme de séquences en proposant une autre lecture et vision des interactions entre agents.

IV.4.1 Architecture et Modélisation du Modèle SMA1

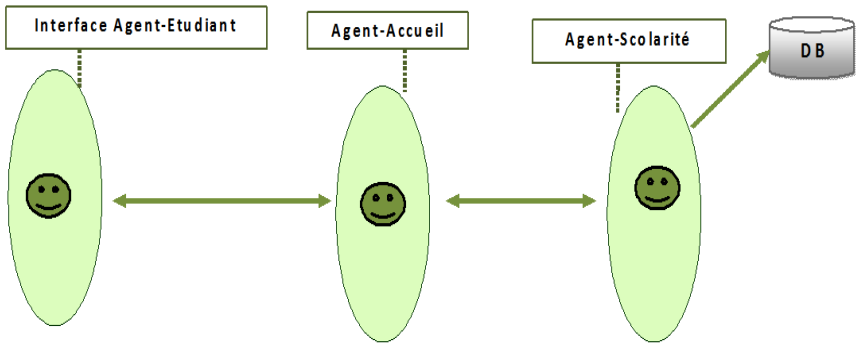


Figure 38: Architecture du modèle SMA1

La figure 38 illustre l'architecture du modèle SMA1 « Scolarité-Intelligente », en présentant une généralité sur l'ensemble des agents acteurs de ce modèle, leurs interactions et la base de données. Les fonctionnalités seront détaillées dans la suite.

Les détails de fonctionnalités de ces agents seront présentés dans ce chapitre par les diagrammes de cas d'utilisations, de classes et d'interactions. En effet, on décrira l'architecture les agents de ce Modèle, leurs communications et coordinations, dont le but de réaliser une application distribuée pour la gestion des demandes d'attestation au sein du service scolarité en utilisant la plateforme multi-agents JADE.

IV.4.2 Diagramme de cas d'utilisation

Ce diagramme de cas d'utilisation présente une description générale des agents du système. Il décrit l'ensemble des fonctionnalités de chaque agent.

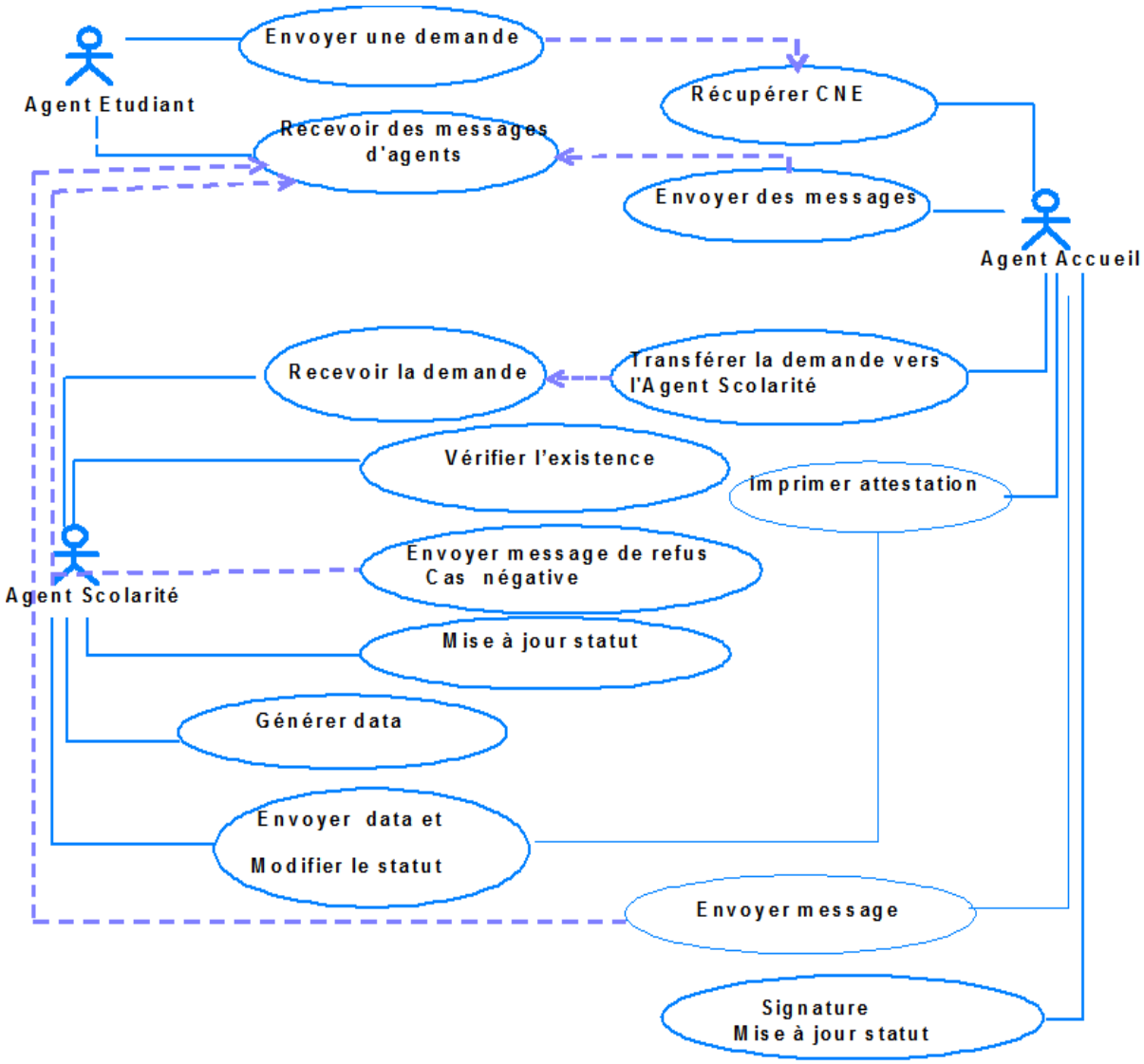


Figure 39: Diagramme de cas d'utilisation SMA1

IV.4.3 Diagramme de classe

Les diagrammes de classes UML (AUML) peuvent être utilisés pour représenter la vision statique des agents.

Nous présentons dans le diagramme de classe (figure 40) la vision statique dans notre application, le diagramme est composée de :

- La classe Etudiant : Elle contient les informations de tous les étudiants qui sont inscrits à l'établissement.

- La classe Demande : Elle stocke toutes les demandes effectuées par les étudiants de l'établissement.
- La classe Statut demande : Elle englobe les statuts de chaque demande : acceptée, en cours, signée…
- La classe Attestation : Elle contient les types d'attestation qu'un étudiant peut demander.
- La classe Filière : Elle contient les filières qui existent dans l'établissement

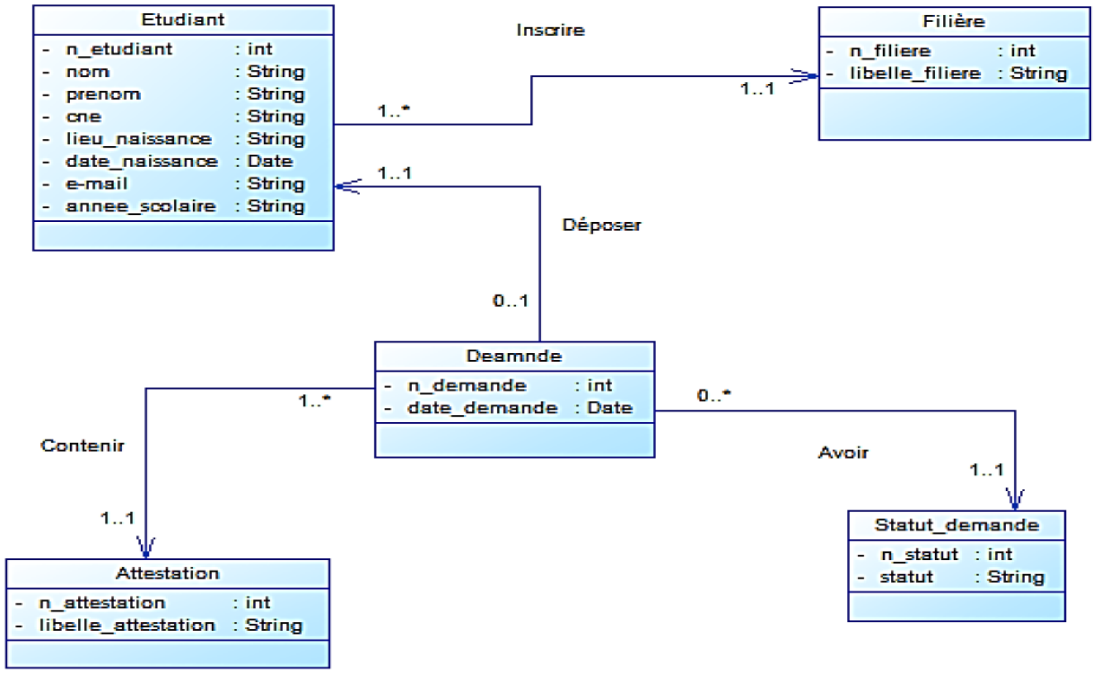


Figure 40: Diagramme de classe

IV.4.4 Diagramme d'interaction :

Le processus de déroulement d'une demande d'attestation dans SMA1.

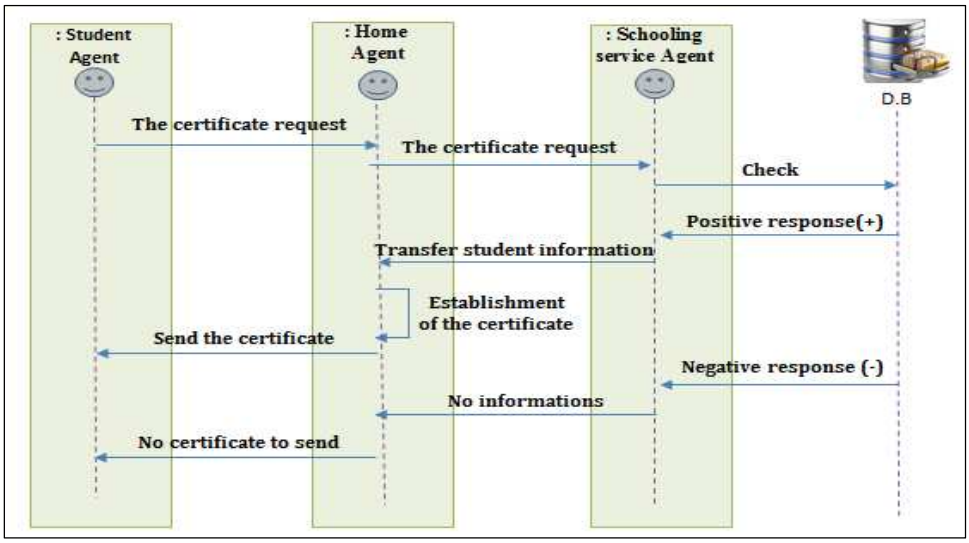


Figure 41: Diagramme d'interactions

Le diagramme d'interactions ci-dessus décrit l'ensemble des stades traversés par une demande d'étudiant dès sa connexion à la plate-forme jusqu'à sa signature par le responsable de avec une précision des tâches réalisée par chaque intervenant sur notre système.

L'agent «**Etudiant**» se connecte à la plateforme, ensuite, il envoie un message ACL à l'agent «**Accueil** », ce message contient le «**CNE**» et le type d'attestation voulue. L'agent« **Accueil** » reçoit ces deux éléments, et les envoie à son tour à l'agent « **Service de Scolarité** ». Ce dernier doit faire une vérification de l'existence de cet étudiant dans la base de données. Si le résultat de la vérification est positif, l'agent « **Service de Scolarité** » récupère les donner de l'étudient et les envoie à l'agent «accueil », qui traite la demande et envoie un message pour la récupération à l'agent étudiants.

IV.5 Description fonctionnelle du modèle SMA1 et résultats

Notre modèle SMA1 est multi-agents. Il est composé de 3 agents, qu'on a créé 3 agents (Agent Etudiant, Agent Accueil et Agent service de scolarité) chacun de ces agents est chargé d'une partie de travail, représentant son but qui est une partie du but global, il coordonne et communique avec les autres agents pour la réalisation de sa mission et par la suite le model SMA1 réalise le but global qu'on a décrit dans les diagrammes ci-dessus.

L'Agent Etudiant est basé sur les fonctionnalités suivantes :

- Bénéficie d'une interface graphique permettant à l'étudiant de saisir ces informations ;
- Un modèle de communication lié à un **Botton** pour envoyer ces informations à l'Agent Accueil et pour interagir avec les autres agents ;
- L'étudiant peut lancer sa demande à partir d'un Smartphone qui supporte le système androïde.

La figure ci-dessous présente cette interface de l' « Agent Etudiant », sert à envoyer une demande, contenant un CNE et le type d'attestation à l' « Agent Accueil ».

Figure 42: Interface de la demande

L' « Agent Accueil » est l'agent principal pour cette plateforme, on a déployé cet agent sur un container sur l'ordinateur central qui contient la main container.

L' « Agent Accueil» agit comme agent intermédiaire entre les deux autres agents, Etudiant et Scolarité.

Dans un autre ordinateur on a programmé un autre agent « Agent Scolarité », dont les tâches est de vérifier la légalité des demandes, et faires des mises à jour successives et automatiques sur la table demande de la base de données, en insérant le statut "encours", "Générée", ou "signée", chaque fois la modification du statut est exécutée.

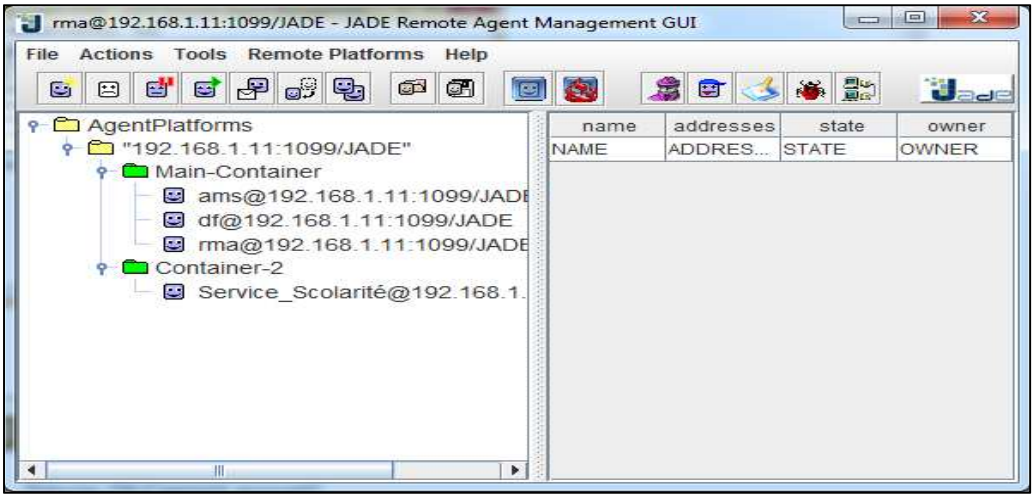


Figure 43: JADE Remote Agent Management GUI

Cet agent utilise une interface d'affichage des interactions passées dans la base de données et affiche les statuts des demandes à chaque instant d'une manière automatique.

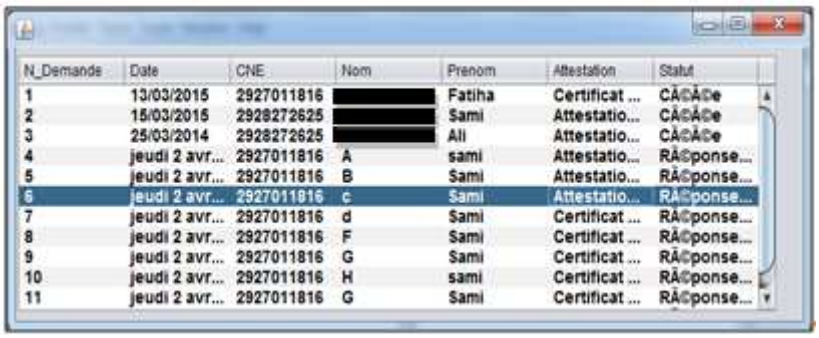


Figure 44: Statut des demandes

Enfin l'«Agent Accueil » crée l'attestation demandée sous forme d'un fichier Excel. Puis, il envoie un message à l'agent «Etudiant» pour lui annoncer que l'attestation est prête.

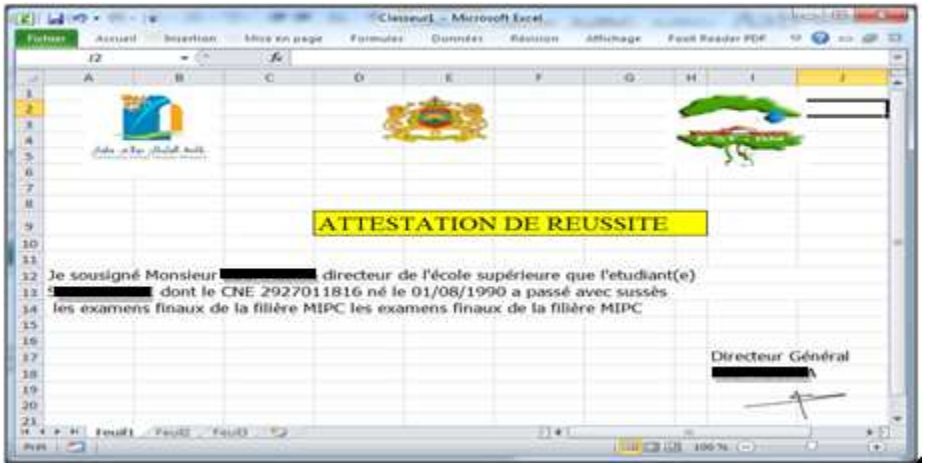


Figure 45: Attestation de réussite générée

IV.6 Conclusion

Nous avons exposé dans ce chapitre les outils utilisés plus la conception de notre projet SMA1, ensuite nous sommes passés aux étapes de spécification, et modélisation que nous avons suivies pour développer cette application multi-agent distribuée en utilisant la plate-forme JADE. C'est un modèle distribuées qui nous a permis de toucher le domaine des SMA, qui est un domaine très complexe, mais aussi très efficace. C'est un travail que nous avons mené en parallèle avec nos recherches, et qui présente un cas réel d'utilisation des SMA, mais ce modèle n'engendre pas toute l'architecture de notre université. Dans le chapitre qui suit, nous allons exploiter les propriétés du dynamisme et évolutivité des SMA pour réaliser un modèle SMA2 plus flexible et adaptable à l'architecture réelle de la scolarité universitaire USMS.

Chapitre V : Modèle SMA2 « Scolarité Intelligente »

Le système d'information de notre université est distribué, le développement d'applications distribuées, ouvertes, et adaptables à l'environnement de son intégration nécessite des moyens qui peuvent fournir des solutions solides et qui sont en mesure de répondre aux exigences. La propriété de distribution impose qu'une application soit implémentée sous forme d'un ensemble d'entités logicielles et qui s'exécutent sur des machines distantes avec une distribution des computations et une décentralisation des ressources et des connaissances.
Nous bénéficions donc particulièrement des avantages des SMA qui sont adaptés aux systèmes complexes, ouverts, distribués et dynamiques pour proposer dans ce chapitre un deuxième modèle SMA2 plus flexible, extensible et adaptable à l'architecture de son système d'intégration. En plus de tous les avantages cités des SMA, nous nous servons aussi de la modularité, qui facilite la conception et la réalisation de ces derniers.

On présente dans cette partie la conception et la mise en œuvre de ce modèle SMA qui réalise un outil d'aide à la gestion des travaux administratifs de Scolarité, nous l'appelons «une Scolarité-Intelligentes ». L'objectif reste le même mais ce dernier Modèle SMA automatise plus de processus scolaire, en intégrant plusieurs entités autonomes (agents) et des agents mobiles qui permettent à ce modèle SMA d'être, évolutif, extensible et adaptable à l'architecture réel de l'université. Ce modèle nous permet de réduire le cout de temps et de personnel et facilitée aux administrateurs leur travail.

V.1 Architecture proposée

Le schéma suivant, montre l'architecture de ce Modèle SMA «Scolarité-Intelligente », un modèle distribué à base des SMA qui nous présente clairement son dynamisme et flexibilité d'adaptation à son environnement d'intégration. Il présente aussi les différents agents utilisés dans ce modèle, les communications et les interactions entre ces agents, ainsi que leurs environnements de déploiement, le détaille de cette architecture est présenté dans ce qui suit.

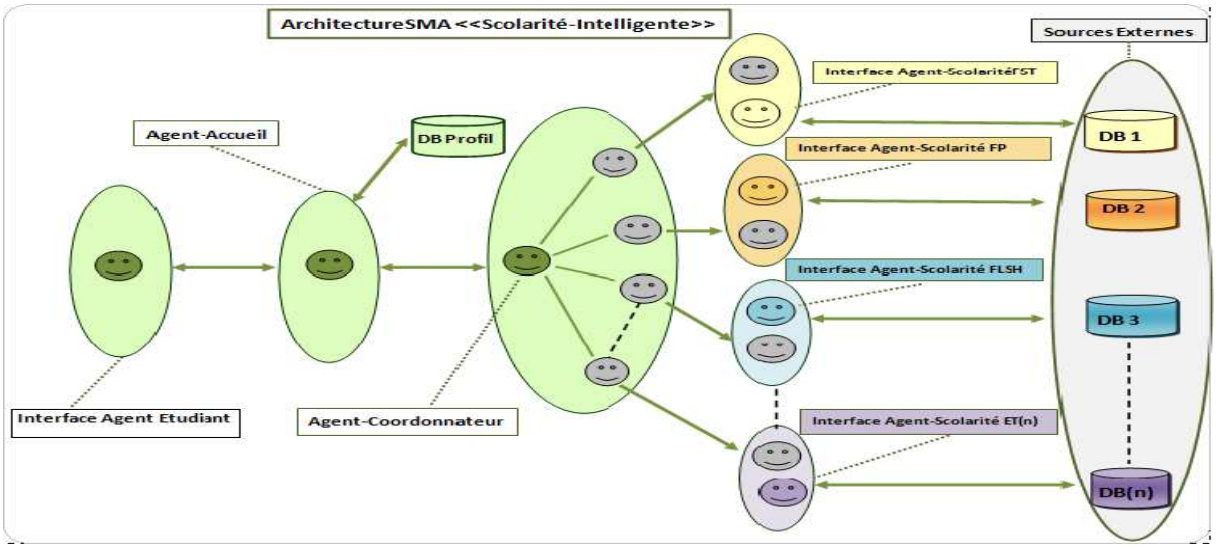


Figure 46: Architecture du Modèle SMA2

Notre modèle SMA2 « Scolarité-Intelligente », est une collection de différents agents persistants, autonomes, coopératifs et mobiles qui opèrent sur un environnement distribué. Les processus de scolarités automatisés, commencent par les demandes des étudiants passant par les vérifications jusqu' à la génération des documents par les agents des scolarités relatifs à toutes les établissements de l'université.

Ce modèle est composé des agents suivants :

- Trois agents stationnaires qui sont : « Agent-Etudiant », «Agent-Accueil » et « Agent-Coordonnateur» ;
- Des « Agent-Scolarité », dont le nombre dépond du nombre des établissements de l'université ;
- Des agents mobiles, un agent mobile par établissement ;
- Chaque agent est représenté dans son environnement de vie « Container »
- Une base de données DB profil qui contient le profil de l'étudiant ;
- Pour chaque établissement une base de données pour enregistrer les demandes.

Ce modèle proposé engendre l'architecture globale du réseau de l'université, qui est distribuée et extensible. La description fonctionnelle détaillée des agents de notre architecture et du système global sera présentée dans les paragraphes qui suivent, dans ce chapitre.

V.2 Exigences Techniques et Outils de développement

V.2.1 Architecture technique

Pour faciliter la réalisation de notre Modèle SMA2 «Scolarité-Intelligente» et le rendre performent, on a besoin de le basé sur une architecture technique puissante.

Dans cette section on va aborder les exigences techniques, sur lesquelles on a basé l'architecture de ce modèle SMA2, les outils utilisés pour sa réalisation et le kit de développement des agents.

L'architecture technique du projet est sous la forme suivante :

- Les données sont stockées dans des bases de données **MySQL**
- L'application se compose de trois couches :
  - La couche **DAO** qui est basée sur Spring Data, **JPA**, Hibernate et **JDBC**.
  - La couche Métier
  - La couche des agents est basée sur le kit **Jade**

Cette architecture est principalement basée sur le « Spring Boot », qui gère les différentes couches, et c'est lui qui se charge de l'injection des dépendances.

Spring : est une plateforme qui se base sur la notion de conteneur léger.

DAO « Data Access Objet » : est la couche chargée de l'accès aux donnés de la bases indépendamment du SGBD.

JPA « Java Persistence API » est une technologie qui a pour objectif d'offrir un modèle « Object Relation Maping). La persistance des données est assurée par le composant JPA et par le « framework Hibernate ».

JDBC « Java DataBase Connectivity » est un pilot qui permet la connexion des applications aux bases de données relationnelles.

V.2.2 Environnement d'implémentation

Les outils utilisés dans l'implémentation de ce projet sont :

- **Spring Tool Suite** (STS) est un **IDE** étendu pour **Eclipse**. Il se spécialise dans le développement des applications à base de l'architecture Spring ;

Jade : le kit de développement des agents

V.3 Modélisation

V.3.1 Diagramme de cas d'utilisation

Le diagramme ci-dessous décrit les grandes fonctionnalités des agents qui constituent ce modèle SMA2 « Scolarité-Intelligente ».

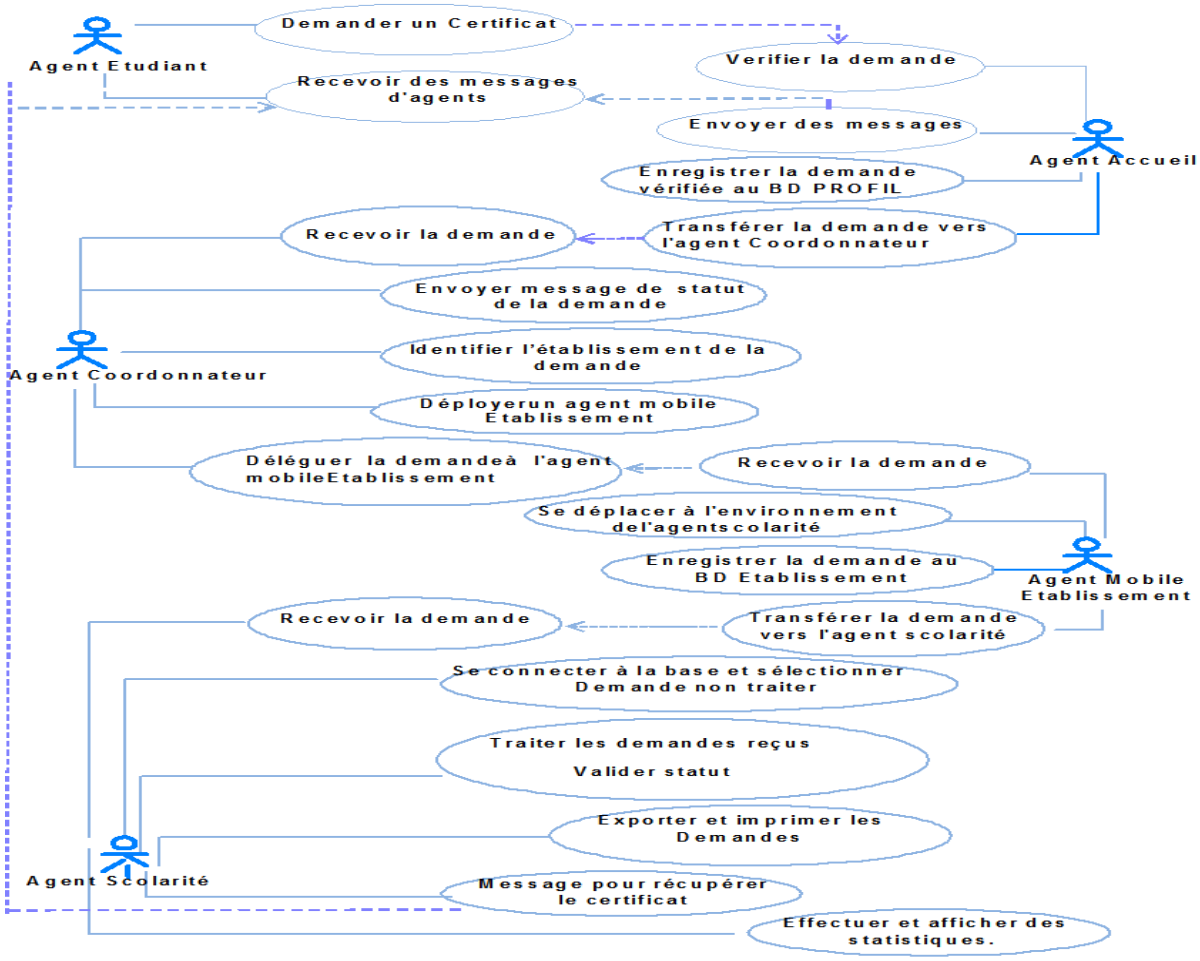


Figure 47: Diagramme de cas d'utilisation SMA2 « Scolarité-Intelligente »

V.3.2 Diagramme d'interaction

Le diagramme ci-dessous présente les interactions entre les agents qui constituent notre modèle SMA2 « Scolarité-Intelligente ».

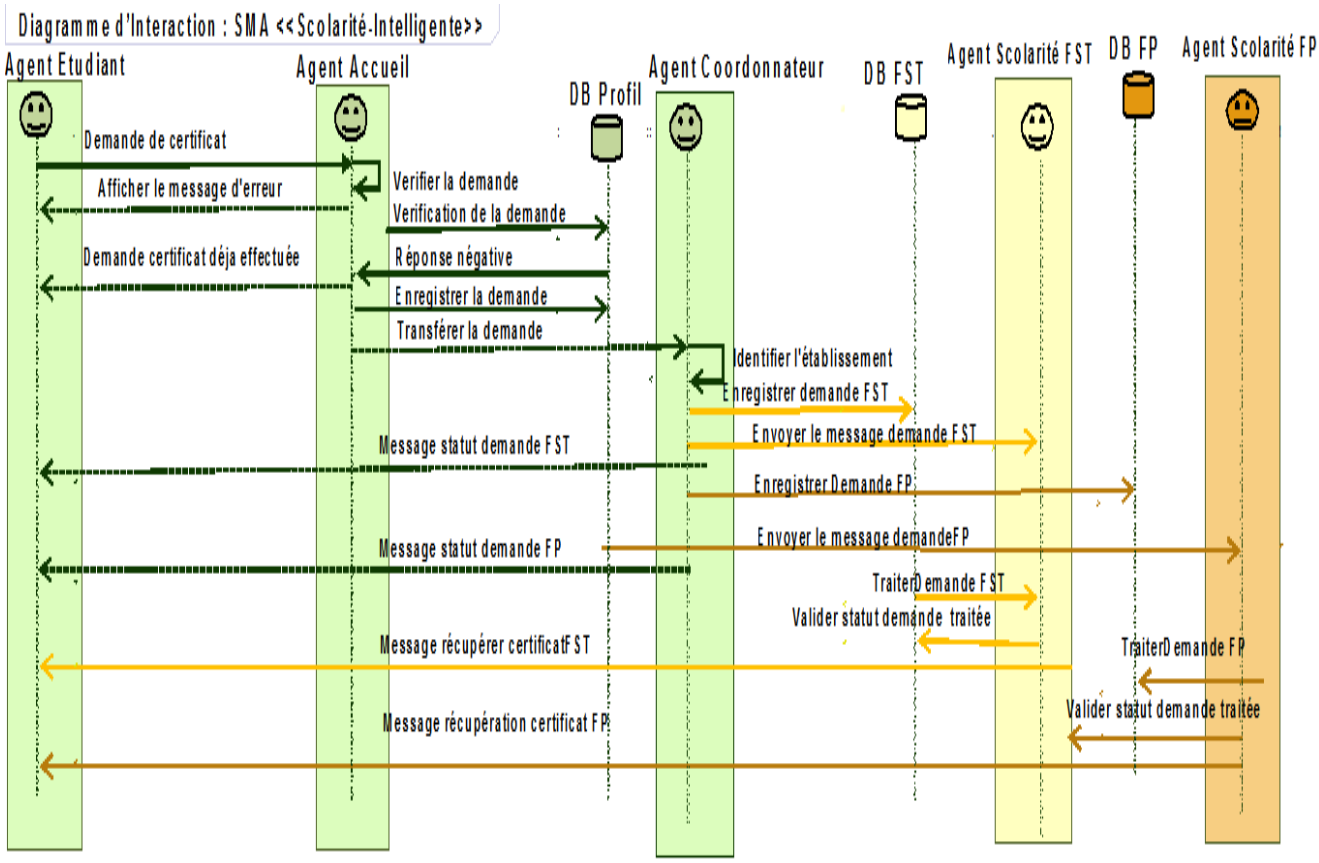


Figure 48: Diagramme d'interaction SMA2 « Scolarité-Intelligente »

Le diagramme d'interaction de la figure 48 présente les différents agents du modèle SMA2 « Scolarité-Intelligente », leurs interactions et leurs communications. En plus il décrit toute les étapes d'une demande, dès le début jusqu'à la récupération de l'attestation. Ce modèle est adaptable à n' importe quel nombre d'établissements , en effet chaque demande effectuée par un étudiant à travers l'interface de l' « Agent Etudiant », passe par tous les étapes des agents « Agent Accueil » et « Agent Coordonnateur », ce dernier identifie le nom de l' établissement et déploie un agent mobile, qui se déplace à l' environnement « container » de l' agent de scolarité correspondant pour lui communiquer la demande et l' enregistrer dans sa base de donner. Donc à partir de l'« Agent Coordonnateur », interviennent des agents mobiles qui se déplacent vers les environnements des établissements, le nombre d'établissement est variable. Nous avons présenté dans ce diagramme seulement deux établissements pour la lisibilité.

Ce diagramme nous montre concrètement l'avantage et la nouvelle solution de modélisations, que les systèmes multi-agent ont apportés. En effet ils offrent la

possibilité de représenter directement les individus, leurs comportements et leurs interactions, ceci et grâce aux agents qui interagissent entre eux de façon autonome et leur modularité.

V.3.3 Diagrammes de classes

Diagramme de classe base de données :

En utilisant la technologie JPA, le diagramme de classe ci-dessous contient les classes qui seront implémentées, ce diagramme est équivalent au diagramme de base de données classiques et il est composée de :

- La classe « Demande Etudiant » stocke toutes les demandes effectuées par les étudiants ;
- La classe « Etudiant » contient les informations de tous les étudiants qui sont inscrits à l'établissement ;
- La classe « Demande » contient tous les informations des demandes effectues avec leur statut d'exécution ;
- La classe « Filière » contient les filières qui existent dans l'établissement
- La classe « Type Demande » contient les types d'attestation à demander.
- La classe « Demande Statut » englobe les statuts de chaque demande(en cours, signée, ..)
- La classe « Etablissement » contient les informations de l'établissement.

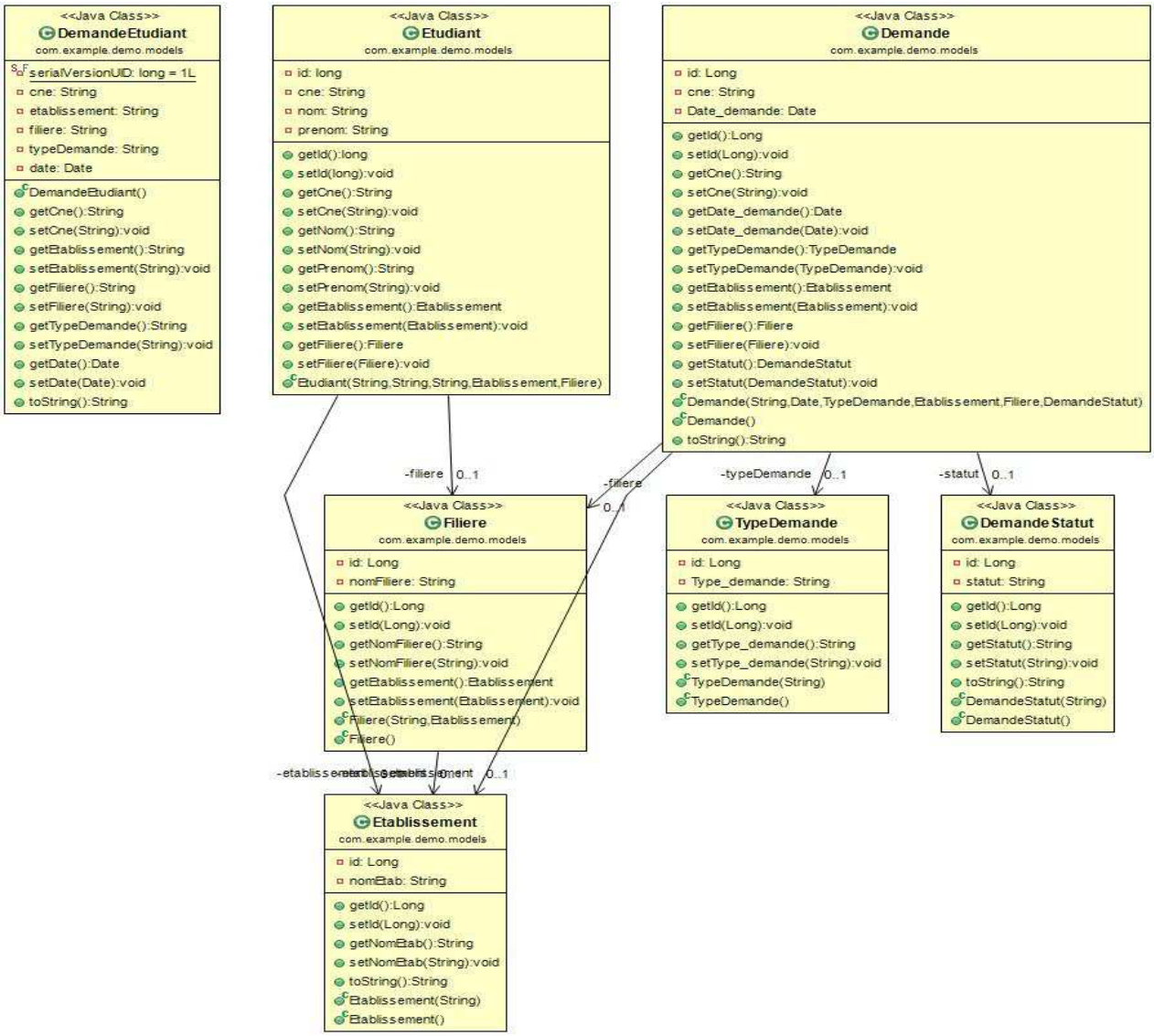


Figure 49: Diagramme de classe bases de données

Diagramme de classe des containers :

Le diagramme de classe ci-dessous présente les différents containers, qui représentent les environnements de déploiements des agents de notre modèle SMA2.

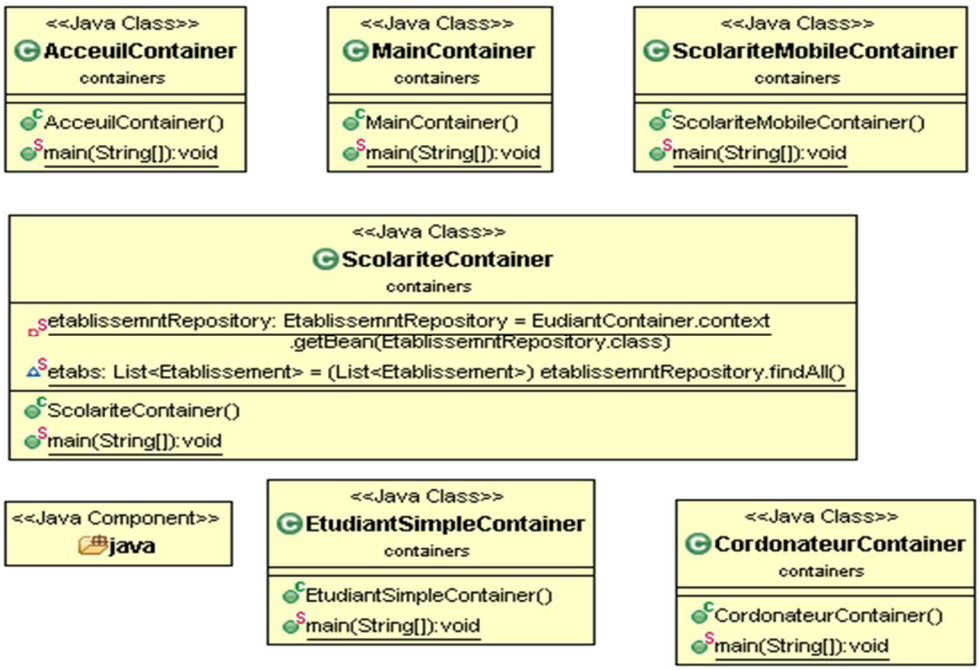


Figure 50: Diagramme de classes des containers

V.4 Résultats et fonctionnement du modèle SMA2

V.4.1 Agents du modèle SMA2

La figure 51 illustre les différents agents de notre modèle SMA2, ainsi que leurs containers. Chaque container représente l'environnement de vie des agents qu'elle contient.

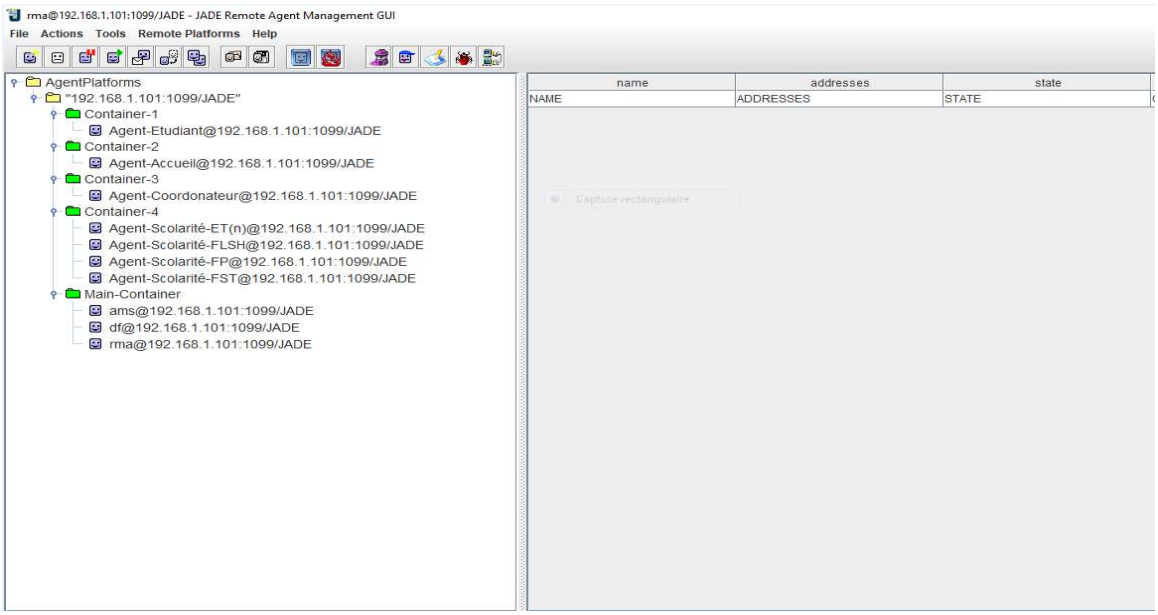


Figure 51: Agents du modèle SMA2 et leurs containers

Les agents sont distribués sur différents containers. Le container 4 héberges les agents des établissements. Dans la réalité chaque agent-Scolarité (établissement), sera déployé dans un container sur un hôte distant. Il suffit d'ajouter l'adresse IP de l'hôte qui héberge ce container pour avoir la communication entres ces containers.

La simulation des agents du modèle SMA2 dans cette plateforme est nécessaire pour pouvoir vérifier la communication entre ces agents dans l'interface graphique de JADE.

V.4.2 Circulation des informations entre les agents du SMA2

La figure 52 illustre la vérification de la circulation des messages entre les agents de notre plateforme, cette communication est faite grâce à l'agent **sniffer** qui permet de suivre les messages échangés dans la plateforme d'agent jade.

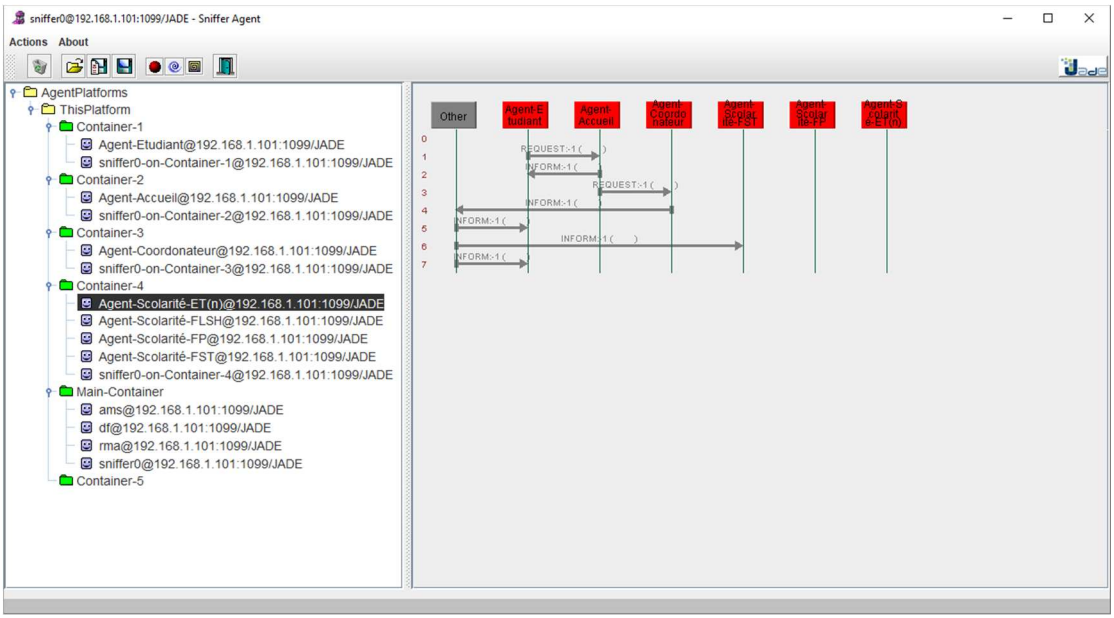


Figure 52: Communication entre les agents du SMA2 dans Jade

Nous avons décrit par les deux figures ci-dessus, les agents de notre modèle, leur environnement de déploiement distribué, qui est présenté par les containers qui les hébergent.

Dans ce qui suit nous allons voir les étapes et les résultats de l'expérimentation d'une demande d'attestation par l'exécution de notre modèle SMA2 « Scolarité-Intelligente ». Ce modèle est dédié à la demande de tous les types des attestations de scolarité. Nous présentons l'attestation d'inscription comme cas d'expérimentation.

V.4.3 Agent étudiant

L'agent étudiant «Agent-Etudiant » possède une interface qui permet à l'étudiant de réagir avec le modèle SMA2 « Scolarité-Intelligente ». Cette interface assure à l'étudiant les taches suivantes :

- Entrer le CNE ;
- Sélectionner l'établissement, la filière et le type de demande ;
- Envoyer la demande ;
- Réception des messages de la part des autres agents.

Les figures ci-dessous illustrent l'interface de l'«Agent-Etudiant » dans différentes étapes :

- Etape1 : Interface initiale.

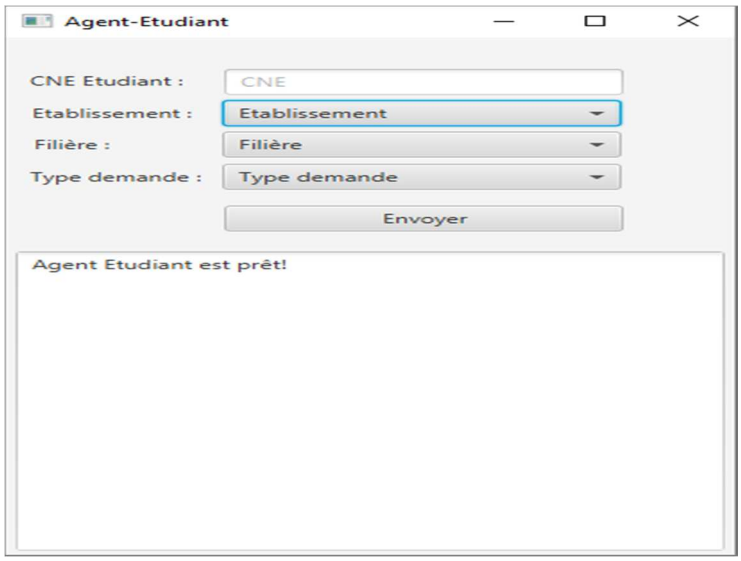


Figure 53: Interface initiale

Cette étape présente l'interface de l' « agent étudiant », avant l'initialisation des paramètres de la demande par l'étudiant.

- Etape 2 : Interface après le remplissage des champs.

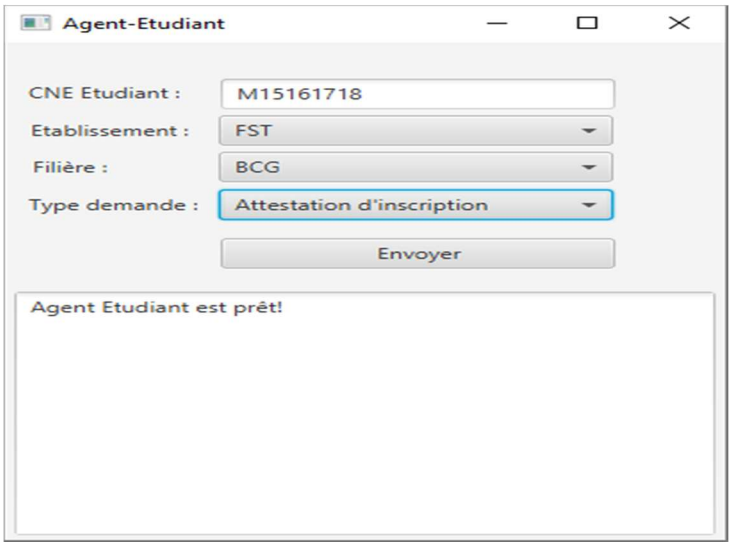


Figure 54: Interface après remplissage des champs

Cette étape présente l'interface de l'« Agent-Etudiant », après le remplissage de tous les champs nécessaires pour effectuer une demande d'attestation d'inscription effectuée.

Pour suivre la communication entre les agents durant les étapes de la demande ; nous allons activer l'agent **sniffer** de Jade. Avant d'envoyer la demande, le diagramme de communication entre les agents est sous la forme suivante.

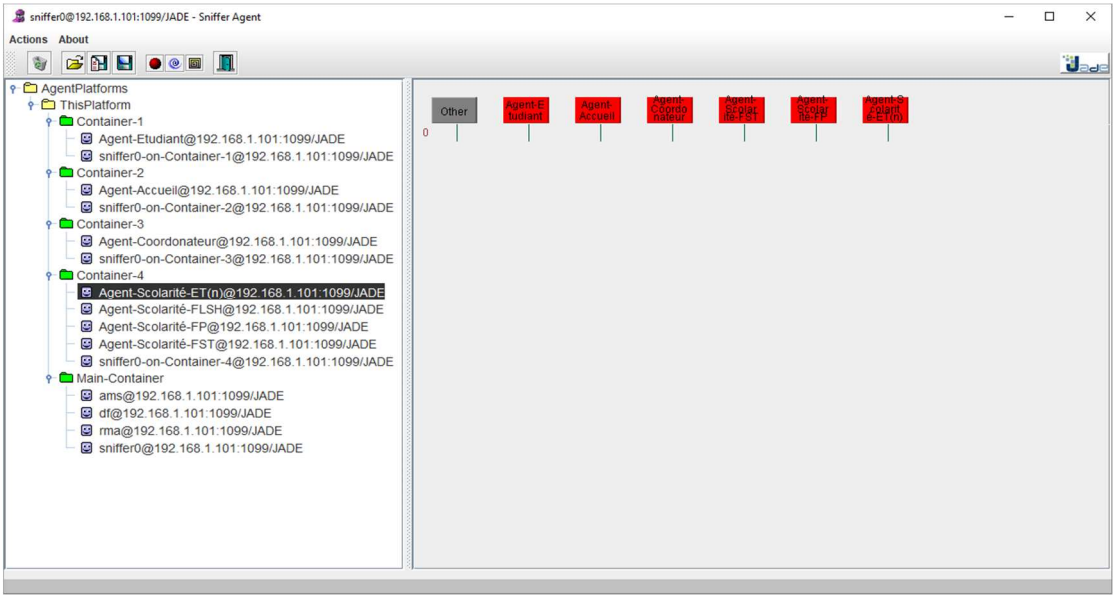


Figure 55: Diagramme de communication avant d'envoyer la demande

L'étudiant envoie sa demande en appuyant le **Botton** « Envoyer », cette demande sera reçue par l' « Agent-Accueil ».

V.4.4 Agent accueil

L'agent accueil « Agent-Accueil », le rôle principal de cet agent est la vérification de la validité de la demande qu'il ait reçue de la part de l'agent étudiant. Cette vérification est effectuée en se connectant à la base de données « DB » profil, qui contient le profil de tous les étudiants qui ont envoyé des demandes.

Cet agent assure les tâches suivantes :

- Vérifier la validité de CNE ;
- Se connecter au DB profil et vérifier la validité de la demande ;
- Envoyer un message à l'agent étudiant dans le cas négatif et stopper le processus de la demande ;
- Enregistrer la demande dans la DB profil le cas positif ;
- Envoyer un message à l'agent étudiant pour l'informer que sa demande est enregistrée ;
- Envoyer la demande à l'agent coordonateur dans le cas positif.

La figure ci-dessous présente le résultat de la demande « M15161718 », lorsqu' elle arrive chez l'agent accueil.

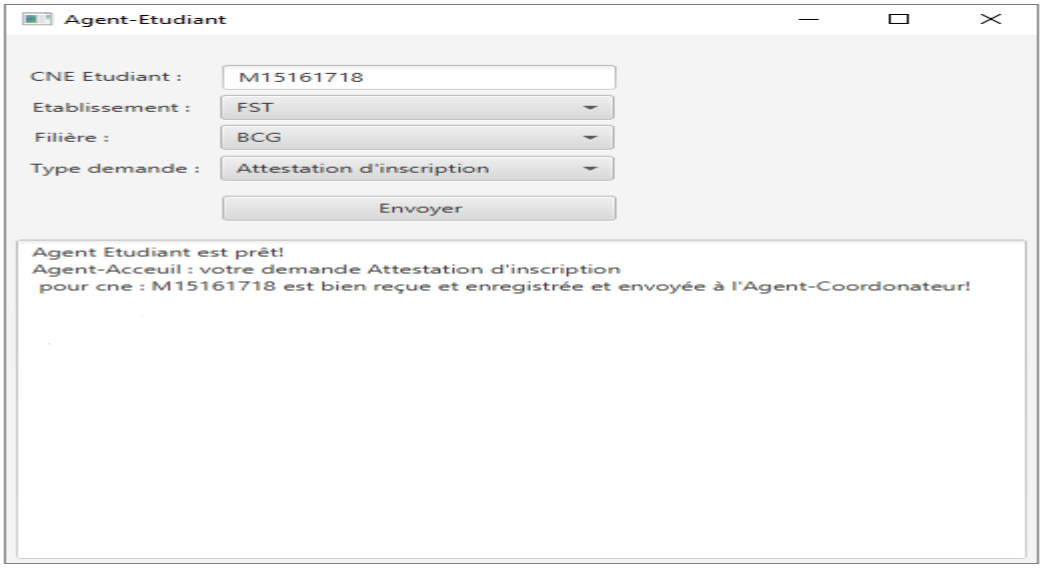


Figure 56 : Réception d'un message de l'agent accueil

V.4.5 Agent coordonateur

Cet agent est responsable de l'identification de l'établissement indiqué dans la demande qu'il a reçue de la part de l'agent accueil.

Les tâches suivantes sont assurées par cet agent :

- Identifier l'établissement de destination ;
- Déployer un agent mobil et lui affecter la demande ;
- Envoyer l'agent mobil ver le container de l'agent scolarité de l'établissement identifier ;
- Envoyer un message du statut de la demande à l'agent étudiant.

Pour chaque demande reçue par l'agent coordonateur, un agent mobil sera déployé pour se déplacer ver le container de l'agent scolarité concerné par cette demande. Dès son arrivée à ce container, l'agent mobil enregistre la demande dans la base de donner, envoie cette demande à l'agent de scolarité qui existe sur ce container et enfin il envoie un message de statut de la demande à l'agent étudient. Puis cet agent va mourir.

La figure ci-dessous illustre l'état de notre demande envoyer par l'agent étudiant. Dans cette étape, l'apparition d'un agent mobile, déployer par l'agent coordonateur ; pour cette demande. Cet agent mobile, hébergé dans le container 5, se déplacera vers l'agent scolarité.

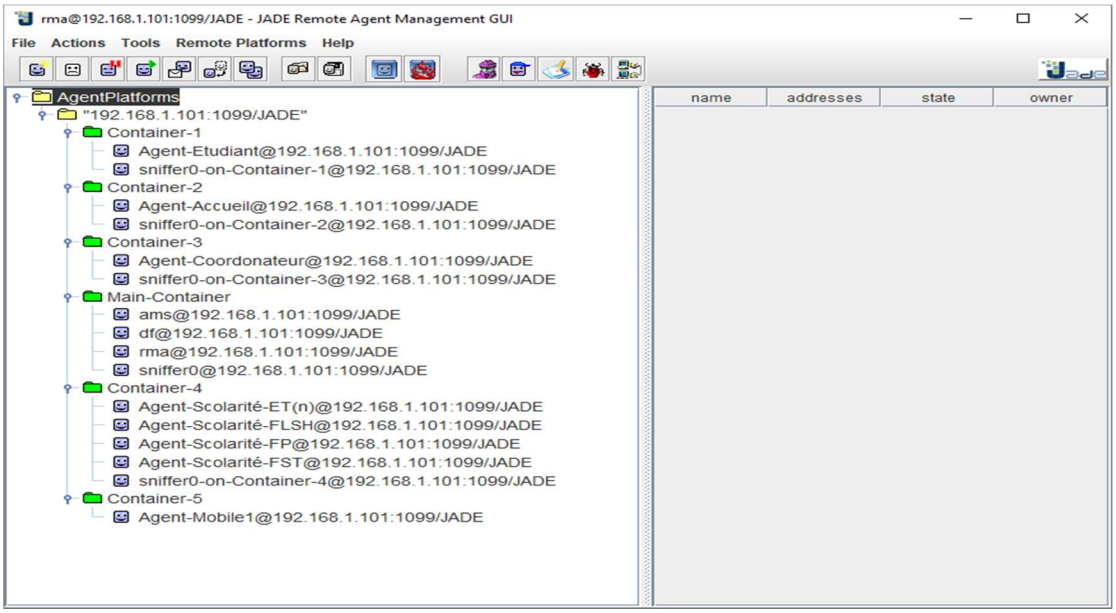


Figure 57: Etape de déploiement de l'agent mobile

V.4.6 Agent scolarité

Cet agent est propriétaire d'une interface graphique principal, qui présente un tableau de bord de ces fonctionnalités. Il est responsable de la gestion des demandes de scolarités. Les principales tâches, qui sont assurées par cet agent :

- Réception des demandes de la part de l'agent mobile ;
- Se connecter à la base de donner des demandes ;
- Traiter les demandes ;
- Envoyer un message du statut de la demande à l'agent étudiant ;
- Imprimer les attestations.

Les figures ci-dessous, décrivent les résultats, lorsque la demande est arrivée au niveau de l'agent scolarité.

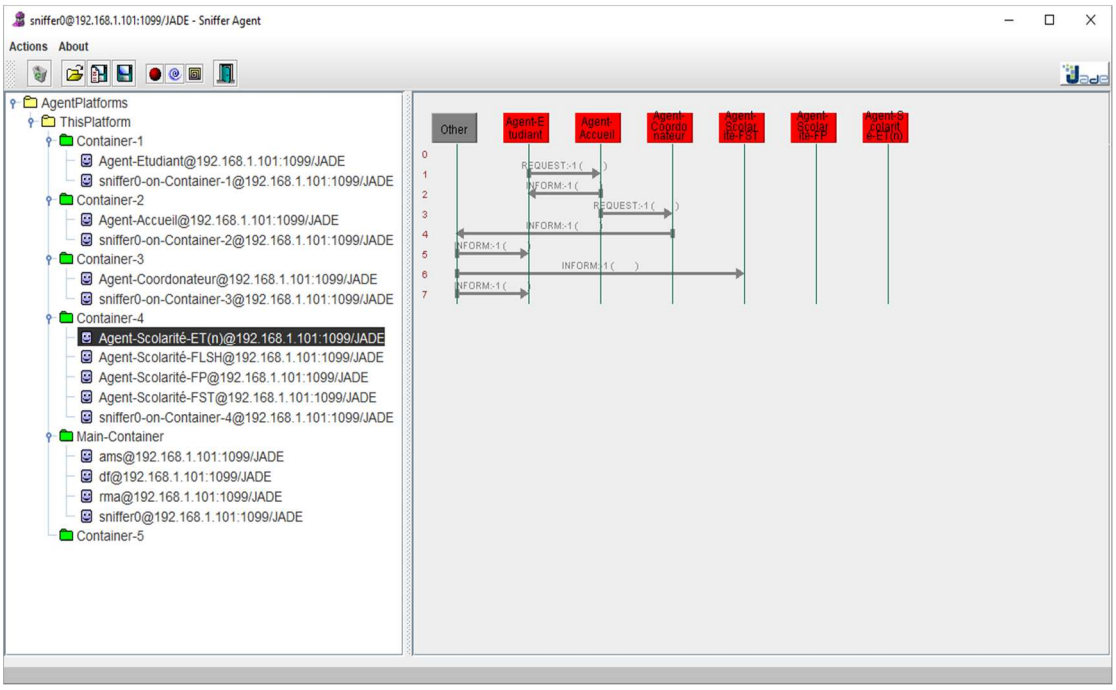


Figure 58: Diagramme d'interaction entres les agents de notre mande

L'agent « other » joue le rôle de l'agent mobil, dans ce diagramme.

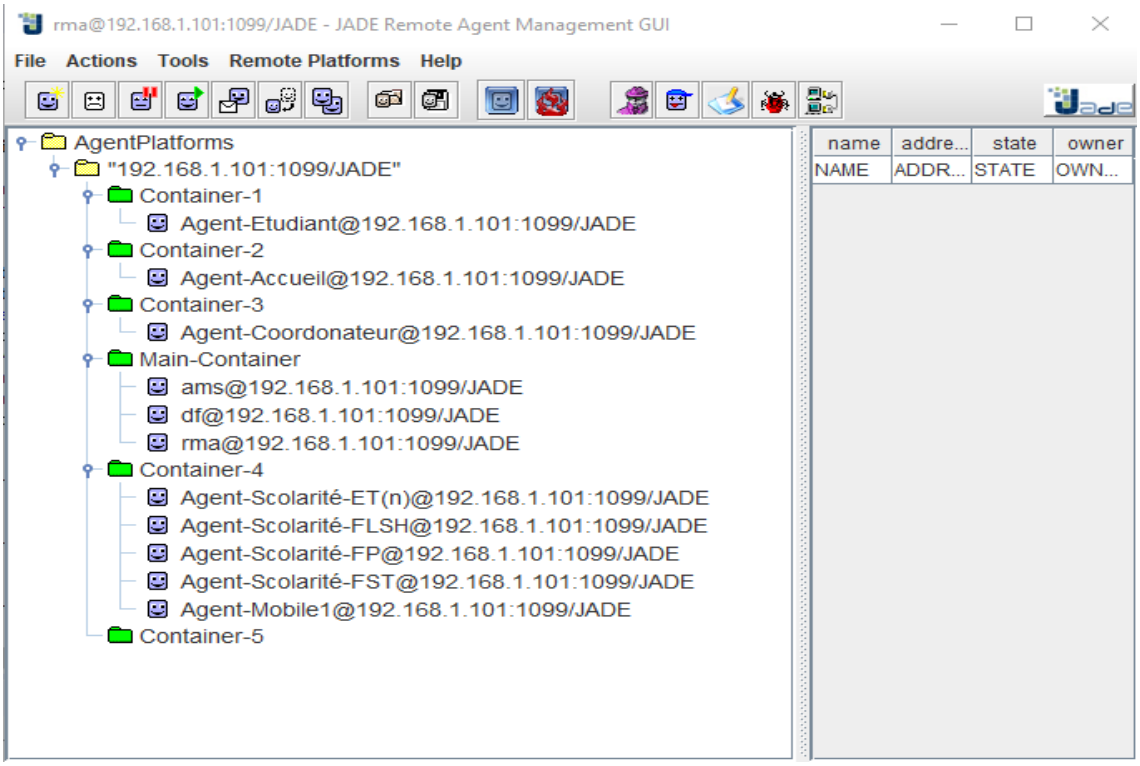


Figure 59: Agent mobile a quitté le container 5

Figure 60: Messages arrivés à l'agent étudiant dans cette étape

L'agent mobile est mort après l'envoie du message, qui apparait dans la figure 60.

La figure ci-dessous présente l'agent scolarité FST, c'est un agent qui possède une interface graphique. Notre demande « M15161718 » est arrivée à cet agent, son statut « Encours de traitement ».

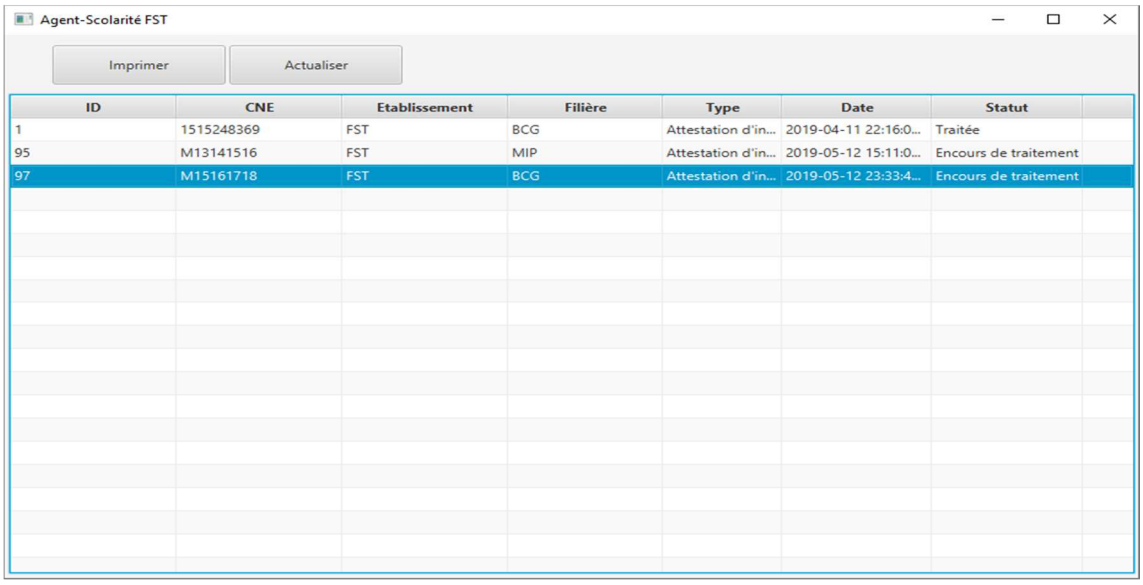


Figure 61: Interface de l'Agent-Scolarité FST

L'agent scolarité FST se connecte à sa base de donner, traite cette demande et enfin l'imprime.

La figure 62, illustre la sélection de la demande « M15161718 », et son impression par l'agent scolarité FST.

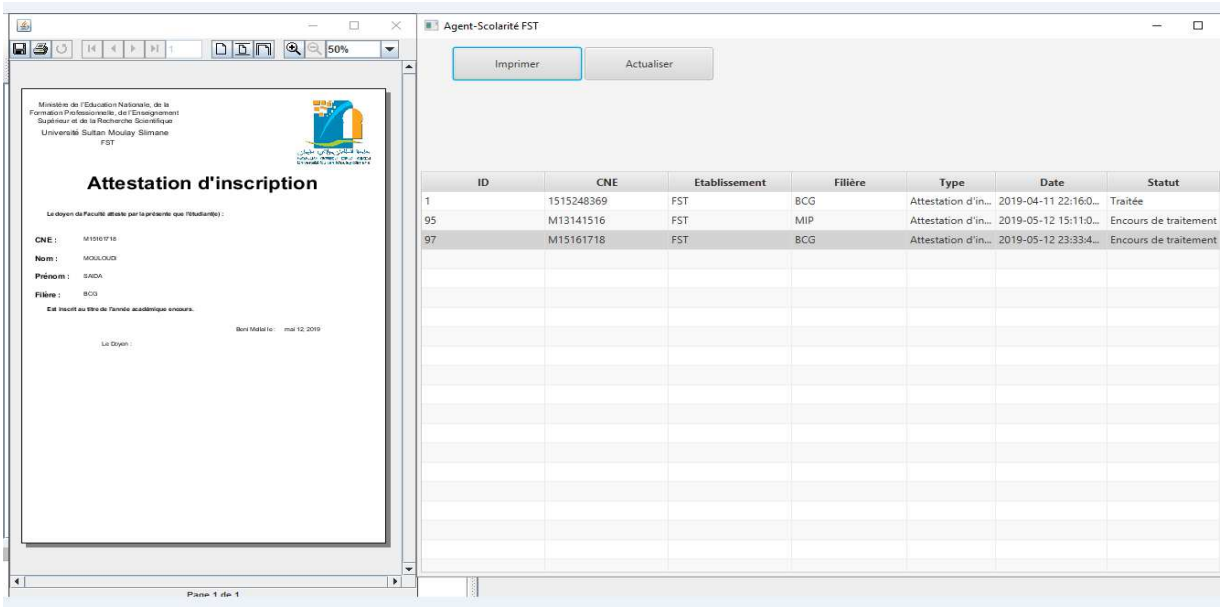


Figure 62: Sélection et impression de la demande

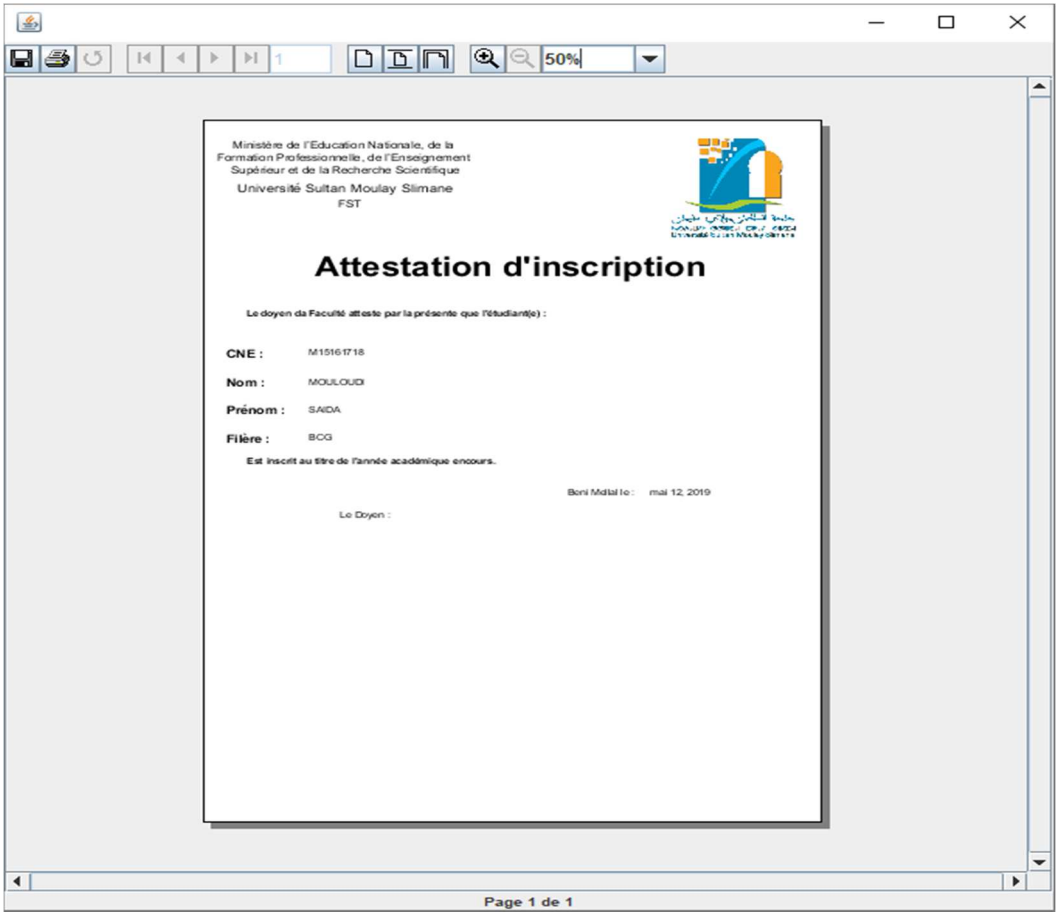
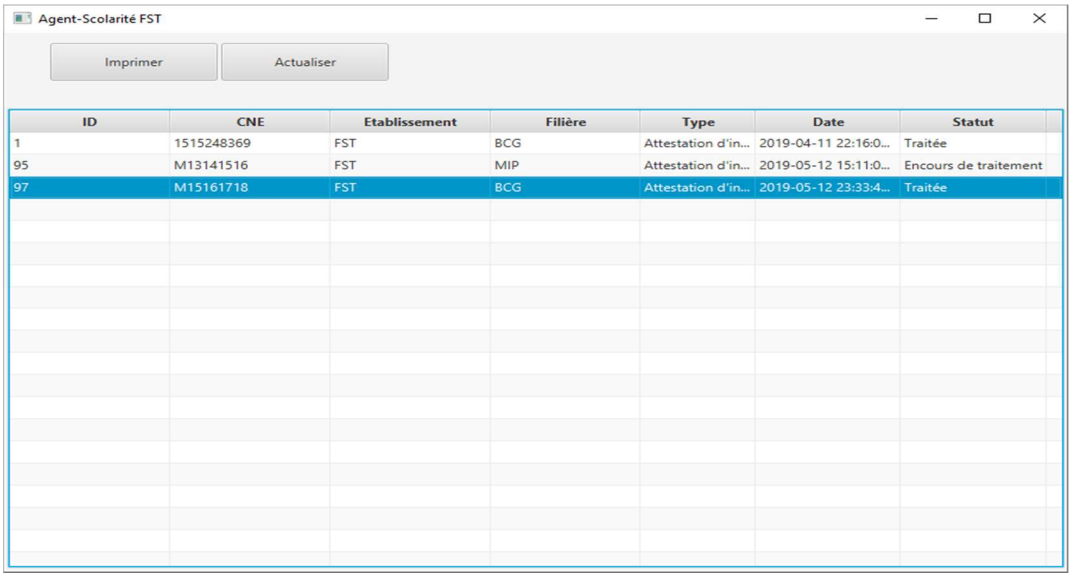


Figure 63: Attestation imprimée

L'agent scolarité FST click le Botton « Actualiser », le statut de la demande passe à « Traitée »



ID	CNE	Etablissement	Filière	Type	Date	Statut
1	1515248369	FST	BCG	Attestation d'in...	2019-04-11 22:16:0...	Traitée
95	M13141516	FST	MIP	Attestation d'in...	2019-05-12 15:11:0...	Encours de traitement
97	M15161718	FST	BCG	Attestation d'in...	2019-05-12 23:33:4...	Traitée

Figure 64: Statut de la demande après son traitement

L’étudiant reçoit le dernier message dans la figure suivante :

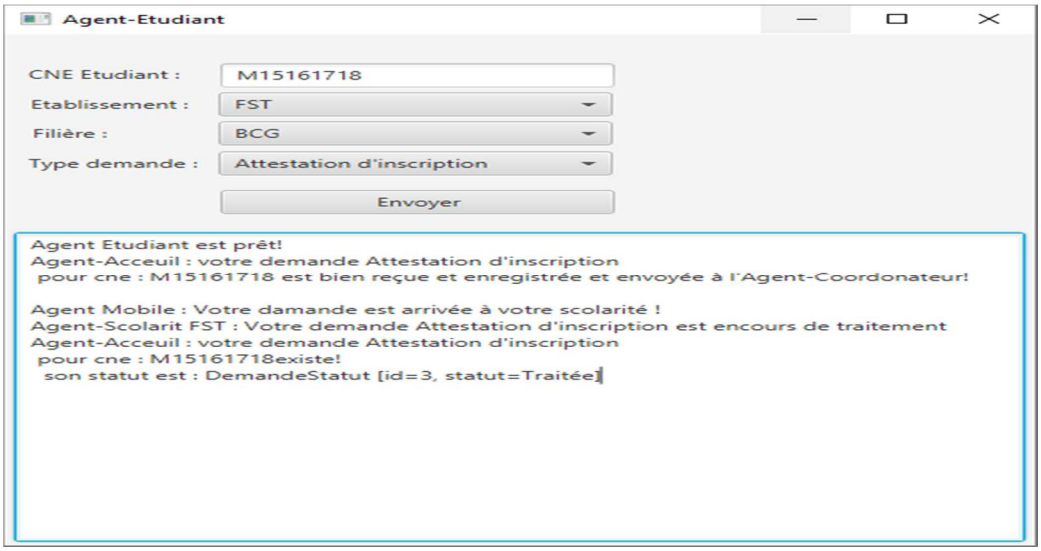


Figure 65: Messages reçues par l’étudiant dans tous les étapes

Une deuxième demande (Etudiant FP).

Pour bien et complètement décrire la fonctionnalité de notre modèle SMA2, nous présentons brièvement la procédure d’une deuxième demande (Etudiant FP). Nous simulons le cas de cette demande, qui est effectuée directement après la demande « M15161718 » (FST).

On présentera ici, seulement la première et la dernière étape, les autres étapes sont similaires à ceux de la première demande.

- **Etape initial** : « Agent-Etudiant », préparation de la demande « z13141516 ».

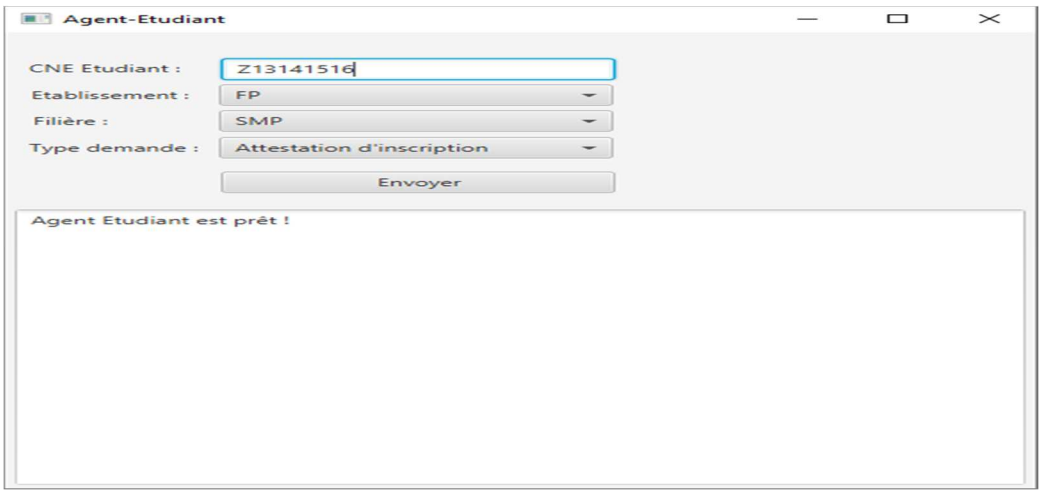


Figure 66: Remplissages des champs d’une demande FP

A ce niveau, le diagramme de simulation des interactions figure 67, est encore dans son état après la première demande FST

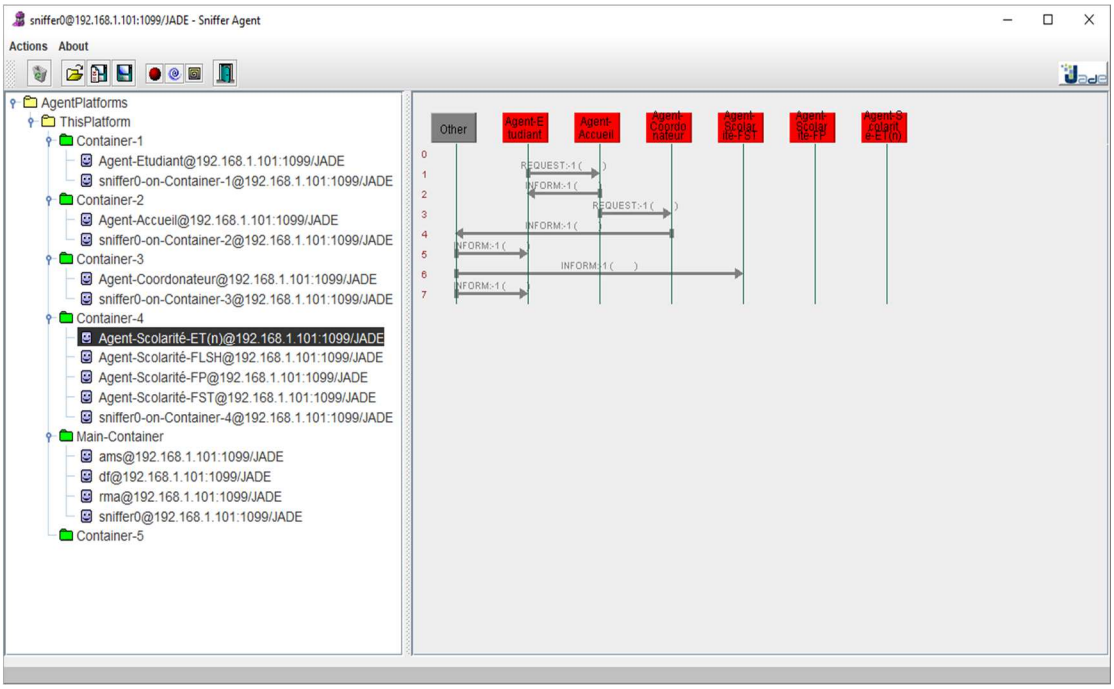


Figure 67: Etat initial du diagramme pour cette deuxième demande

-**Etape final** : La demande est arrivée à l' « Agent-Scolarité FP»

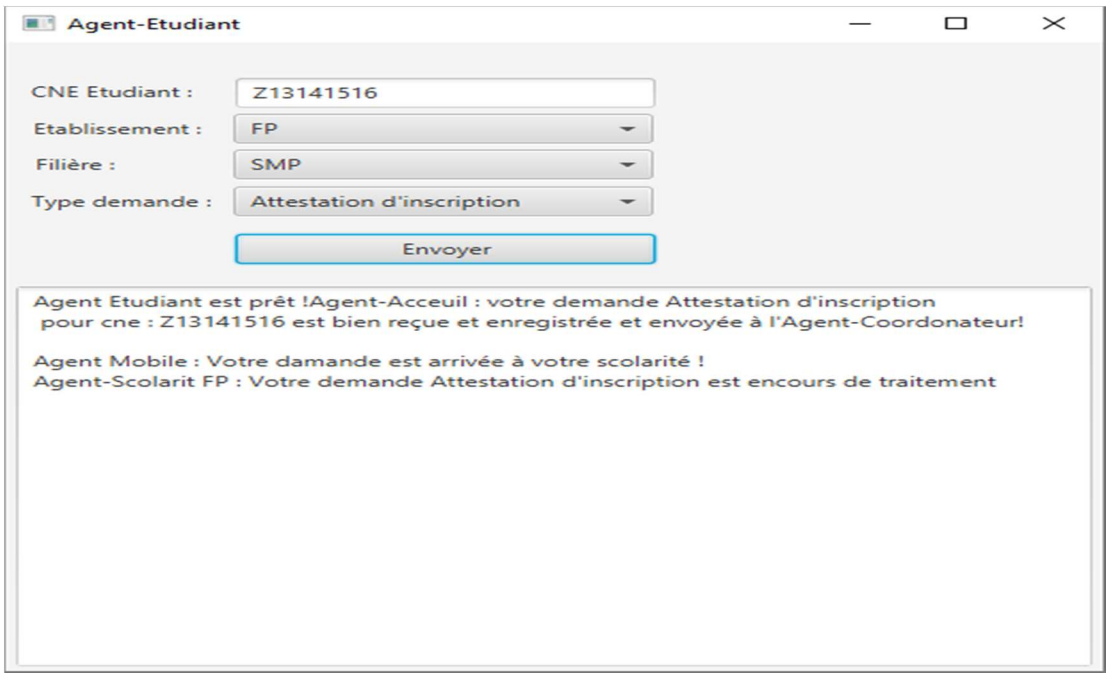


Figure 68: Messages envoyés

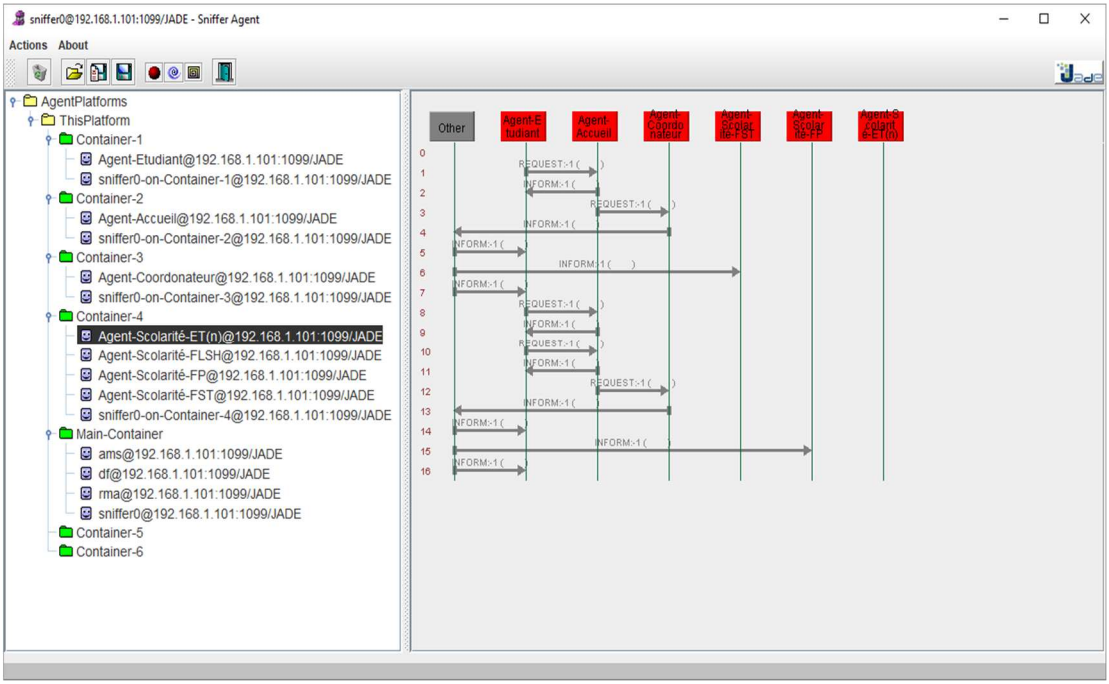
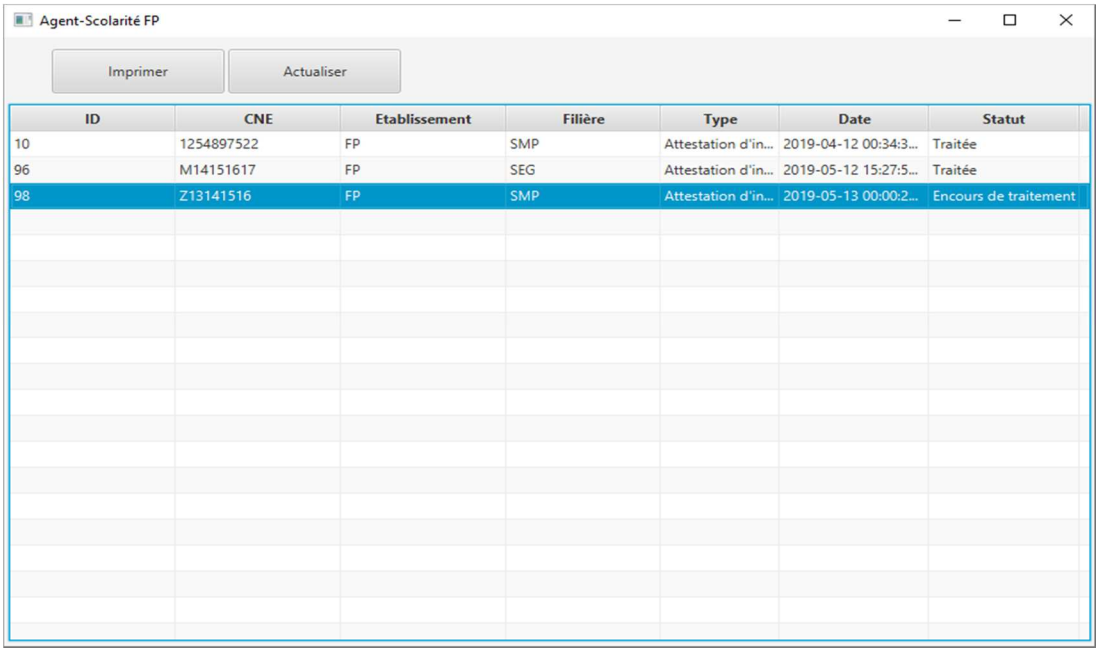


Figure 69: Diagramme d'interaction après la deuxième demande



ID	CNE	Etablissement	Filière	Type	Date	Statut
10	1254897522	FP	SMP	Attestation d'in...	2019-04-12 00:34:3...	Traitée
96	M14151617	FP	SEG	Attestation d'in...	2019-05-12 15:27:5...	Traitée
98	Z13141516	FP	SMP	Attestation d'in...	2019-05-13 00:00:2...	Encours de traitement

Figure 70: Agent-Scolarité FP

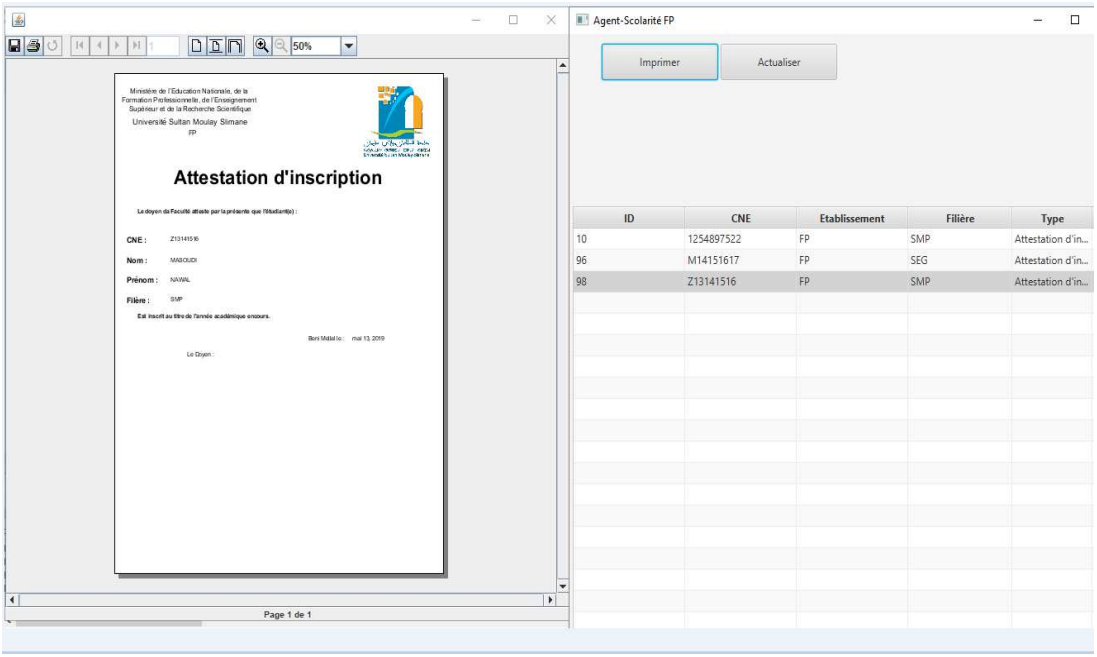


Figure 71: Impression de l'attestation

On peut évoluer l'interface de l'agent scolarité, en ajoutant d'autres fonctionnalités. Son interface présente donc un tableau de bord évolutif sans toucher les autres agents. Pour aider cet agent de scolarité à faires d'autres taches dont il a besoin, exemples des statistiques ou autre, On peut implémenter d'autres agents séparément, leurs attribuer des tâches et les connecter à son interface graphique par des Bottons, qui lancent leurs déploiements, sans changer l'architecture de notre modèle SMA2. Cette évolutivité est un bénéfice de la propriété modularité des Systèmes multi-agents.

Ce modèle SMA2, est dynamique, évolutif et extensible. En effet, on peut ajouter le nombre d'agents de scolarité, qu'on veut sans toucher l'architecture du modèle.

V.5 Conclusion

Ce modèle SMA2, nous a permet de voir concrètement les résultats de l'analyse comparative, que nous avons effectué dans le deuxième chapitre de ce mémoire. La distribution à base de l'architecture 3-tiers que nous avons réalisé dans l'application (objet, JEE), reste limitée.

Le modèle multi-agent avec son faible degré de couplage et son haut niveau d'abstraction, qui lui offrent un fort dynamisme et modularité, permet d'approcher le modèle à la réalité, ainsi, il permet d'effectuer une simulation de l'état comme s'il s'agissait d'un laboratoire virtuel ou chaque agent peut communiquer, se déplacer,

interagir et agir suivant les changements de son environnement. Et le plus important est que chaque agent est marqué comme pourrait être un être naturel car il peut être suivi à tout moment pendant sa période de vie, les agents sont distingués les uns des autres et l'ajout ou la retrait d'un ensemble d'agents est aisé du fait qu'ils sont modulaires.

Une autre caractéristique importante de la modélisation Multi-agents, réside dans son incrémentalité et évolutivité. En effet chaque agent est décrit par son propre code que l'on peut facilement modifier sans toucher le modèle global.

Conclusion générale et perspectives

Dans le cadre de cette thèse nous nous sommes intéressés à l'ingénierie des systèmes multi-agents au paradigme objet et son évolution, et par la suite au développement d'une application distribuée à base de l'architecture 3tier qui modélise les paradigmes objet et deux modèle distribuées SMA. Ces applications à bases des paradigmes objet et agent ont pour cas d'études la scolarité Universitaire.

Le point de départ de nos travaux a été de parcourir l'état de l'art relatif aux techniques d'ingénierie des systèmes distribués en se basant sur les systèmes multi-agents et le paradigme objet.

Nous avons classé ces travaux en grandes familles :

- Une généralité sur l'évolution de l'intelligence artificiel (IA)vers l'intelligence artificiel distribuée (IAD), qui a prouvé son utilité et son efficacité en permettant de développer des logiciels dynamiques pour résoudre le problème de distribution des applications et par suite dépasser le monde des logiciels statiques et monoposte (basés dans une seule machine isolée).
- Etudes des SMA consacrée sur :
  - langages spécifiques au paradigme agent qui facilitent la spécification des systèmes multi agents.
  - méthodologies orientées agent qui guident les développeurs durant les phases de développement
  - Les plateformes multi-agents qui facilitent l'implémentation et le déploiement des systèmes multi-agents.-
- L'évolution des paradigmes objet ainsi que les méthodologies qui sont relatives à ces paradigmes et qui ont données naissances à des nouveaux outils souples et faciles, utilisée dans le domaine de développements des applications distribuées.

Suite à ces études, on a conduit une analyse comparative entres ces différents paradigmes. Cet analyse est restrictif au côté structurel, qui est identifié par l'ensemble des concepts tangibles au niveau du modèle, nous avons analysé la hiérarchisation en se basant sur une perspective générale d'évolution de la programmation et le dynamisme en fonction du couplage structurel des entités.

En générale les études effectuées dans cette partie théoriques nous ont révélé de grandes disparités relatives à l'application des théories orientées objet et agent. Ainsi, nous avons constaté que :

- différents concepts ont été définis et utilisés au niveau des méthodologies, langages de modélisation et plates-formes d'implémentation. La sémantique de ces concepts est souvent abordée de manière informelle, ce qui a suscité différentes interprétations au niveau des différentes techniques d'ingénierie.
- Plusieurs concept et approche ont été spécifiées pour la modélisation des SMA. Nous avons conclu que l'utilisation d'un concept plutôt que l'autre dépend entièrement du domaine d'application.

Après avoir exposées ces études théoriques, nous sommes passées à la phase de conception et de mise en œuvre des applications objectives qui représentent un service de scolarité online pour automatiser le service de scolarité de l'université. Ces applications sont indépendante l'une de l'autre et ont le même but. La première est à base de JEE qui représente une plateforme pour le développement des applications distribuées orientées objet. Les deux autres Modèles sont des SMA à base des agents.

D' après ce travail nous avons conclu que le développement des applications distribuées à base des objets nécessite plus en termes de temps et d'outils indispensables pour la réalisation de ces applications, qui restent limitées dans l'adaptation et l'évolutivité. Malgré ces difficultés qui se présentent dans la conception et la réalisation des modèles SMA, ils ont des avantages majeurs, liées à leurs flexibilités et dynamismes dans la distribution.

Nous proposons comme perspective pour l'amélioration de ce travail dans le futur, la réflexion sur les possibilités d'étude et de réalisation d'une approche qui combine les agents et les composants logiciels, pour avoir des résultats plus efficaces et moins couteuses.

BIBLIOGRAPHIE

[1] Jennings, N. R., Wooldridge, M.: Agent-Oriented Software Engineering. In Handbook of Agent Technology (ed.) Bradshaw, J. AAAI / MIT Press, 2000 (to appear).

[2] A.H. BOND et L.GASSER Reading in distributed artificial intelligence, mogan Kaufmann publishers, Inc 1988.

[3] V. R. Lesser and D. D. Corkill. Distributed problem solving networks.
John Wiley and Sons, New York, 1987.

[4] V. R. Gasser, E. H. Durfee, and D. D. Korkill. Coherent cooperation among communicating problem solvers. IEEE Transaction on Computers, 36, 19, 1987.

[5] R. Davis. Report on the workshop on distributed artificial intelligence. In SIGART Newsletter, page 42-52, janvier1980.

[6] CORKILL D. D. and LESSER V. R., "The Distributed Vehicle Monitoring Test bed: atoll for investigating Distributed Problem Solving Network", AI Magazine 3(4), pp. 15-33, 1983.

[7] J. S. ROSENSCHEIN and GENESERETH M. R., "Deals Among Rational Agents", 84-44HPP Report, Stanford University, 1984.

[8] GASSER L., "Social conceptions of knowledge and action: DAI foundations and open systems semantics", in AI journal, volume 47, pp. 107-138, 1991.

[9] BOURON T., "Structures de Communication et d'Organisation pour la coopération dans un Univers Multi-Agents", Thèse de doctorat, Paris VI, Laforia, 1993.

[10] DROGOUL A., "De la simulation multi-agents à la résolution collective de Problèmes", Thèse de doctorat, Université de Paris 6, Novembre 1993.

[11] LE STRUGEON E., "Une méthodologie d'auto-adaptation d'un système multi-agents cognitifs", Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambrésis,
France, 1995.

[12] BAEIJS C. et DEMAZEAU Y., "Les Organisations dans les Systèmes Multi-Agents",4èmes Journée Nationale du PRC-IA sur les Systèmes Multi-Agents, Toulouse, France, 1996.

[13] SICHMAN J. S. et CONTE R., "On Personal and Role Mental Attitudes: A preliminary Dependence-Based Analysis", in SBIA'98, Porto Alegro, Brésil, 1998.

[14] Yves Demazeau. From interactions to collective behaviour in agent-based systems. In: Proceedings of the 1st. European Conference on Cognitive Science. Saint-Malo, pages 117–132, 1995.

[15] Ferber J., Les systèmes Multi-Agents Inter Edition, Paris, France, 1995.

[16] Michael Wooldridge, Nicholas R. Jennings, " Intelligent Agents: Theory and Practice", Manchester Metropolitan University, United Kingdom, Submitted to Knowledge Engineering Review, October 1994.Revised January 1995.

[17] Shoham, Y. (1993). Agent-oriented programming. Artificial intelligence, 60(1), 51-92.

[18] Durfee E. H. and Lesser V. R., Using partial global plans to coordinate distributed problem solvers. In Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87), Milan, Italy, 1987.

[19] Rao, A. S. and Georgeff, M. P. (1992). Social plans: Preliminary report. In Werner, E. and Demazeau, Y., editors, Decentralized AI 3 - Proceedings of the Third European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-91), pages 57-76. Elsevier Science Publishers B.V.: Amsterdam, the Netherlands. Rao and Georgeff, 1993.

[20] M.E. Bratman, D.J. Israel, and Martha E. Pollack. Plans and resource-bounded practical reasoning. Computational Intelligence, 4(4):349–355, 1988.

[21] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87), pages 677–682, Seattle, 1987.

[22] M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dMARS. In Agent Theories, Architectures, and Languages, pages 155–176, 1997.
[23] M. E. BRATMAN. Intention, Plans, and Practical Reason.1987.

[24] A. Drogoul. De la simulation multi-agents à la résolution collective
de problèmes. Une étude de l'émergence de structures d'organisation dans les systèmes multi-agents. PhD thesis, University of Pierre et Marie Curie (Paris VI), 1993.

[25] B. Chaib-draa and P. Levesque. Hierarchical models and communication in multi-agent environments. In Proceedings of the Sixth European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-94), pages 119-134, Odense, Denmark, August 1994.

[26] W. Duch, R.J. Oentaryo et M. Pasquier. Cognitive Architectures: Where do we go from here? In Proceeding of the 2008 conference on Artificial General Intelligence, pages 122–136. IOS Press, 2008.

[27] J. Müller and M. Pischel. An architecture for dynamically interacting agents. In CoopIS, pages 114–121, 1994.

[28] O. Boissier and Y. Demazeau (1996). ASIC: An architecture for social and individual control and its application to computer vision. In John W. Perram and Jean-Pierre Müller, editor, Distributed Software Agents and Applications, 6th European

Workshop on Modelling Autonomous Agents – MAAMAW, 1994, volume 1069, pages 1–18, Denmark. Springer.

[29] M. Rodriguez. Modélisation d'un agent autonome : Approche constructiviste de l'architecture de contrôle et de la représentation de connaissances. PhD thesis, Université de Neufchâtel., 1994.

[30] M. Occello and Y. Demazeau (1998). Modelling decision-making systems using agents for cooperation in real time constraints. In third IFAC Symposium on Intelligent Autonomous Vehicles, volume 1, pages 51–56, Madrid, Spain.

[31] D.B. Lange and M. Oshima. Programming and Deploying Java Mobile Agents with Aglets. Addison Wesley, 1998.

[32] V. Chevrier. Contributions au domaine des systemes multi-agents. Technical report, Université Henry-Poincarré - Nancy 1, 2002.
URL http://www.loria.fr/~chevrier/Publi/HDR.zip.

[33] B. Chaib-draa. Interaction between agents in routine, familiar and unfamiliar situations. International Journal of Intelligent and Cooperative Information Systems, 1(5):7-20, 1996.

[34] M.S. Fox M.S. An organizational view of distributed systems. IEEE Trans. Syst. Man. Univ. Cybern , vol. SMC-ll; 1981, pp. 70-80.

[35] F. Zambonelli, N.R. Jennings and M. Wooldridge (2000), «Organisational Abstrations for the Analysis and Design of Multi-Agent Systems» Proc. Ist Int. Workshop on Agent-Oriented Software Engineering, Limerick, Ireland, 127-141.

[36] D. Weyns and T. Holvoet. Architecture-centric software development of situated multi agent systems. In Proceedings of the 7th international conference on engineering societies in the agents world VII, ESAW'06, pages 62–85, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-75522-5, 978-3-540-75522-7.

[37] S. Pest, C. Brassac et P. Ferrent, Ancrer les agents cognitifs dans l'environnement. In : Actes des Se Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents. - La Colle sur Loop, Avril 1997.

[38] O. Boissier, S. Gitton, and P. Glize. Caractéristiques des systèmes et des applications. In y. Demazeau, editor, Systems multi-agents, volume 29 of ARAGO, pages 25–54. Editions TEC DOC, février 2004. OFTA.

[39] M. Le Bars, J.-M. Attonaty and S. Pinson. An agent-based simulation for water sharing between different users. In International Congress, August 28-31, 2002. Zaragoza, Spain (2002).

[40] M. Ljunberg and A. Lucas. The OASIS air traffic management system. In In Proceedings of the Second Pacific Rim International Conference on AI (PRICAI-92), Seoul, Korea, 1992.

[41] B. Esfandiari and D. Quinqueton. An interface agent for network supervision. In Proceedings of ECAI, Budapest, 1996.

[42] G. Booch. Conception orientée objets et applications. Addison-Wesley, 1992.

[43] Clemens Szyperski. Component Software: Beyond Object-Oriented Programming. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[44] George T. Heineman. "An Evaluation of Component Adaptation Techniques"; Rapport technique WPI-CS-TR-99-04, Computer Science Department Worcester Polytechnic Institute, Worcester, MA, USA, 2000.

[45] F. DeRemer and H. Kron. Programming-in-the large versus programming-in-the-small. In ACM Press, editor, In Proceeding of the International Conference on Reliable software, pages 114–121, New York, NY, USA, 1975.

[46] G. Booch, J. Rumbaugh, and I. Jacobson. Le guide de l'utilisateur UML. 2000.

[47] IEEE Architecture Working Group. Recommended practice for architectural description of software-intensive systems. Technical report, IEEE, 2000.

[48] Dewayne E. Perry and Alexander L. Wolf. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes, 17(4), pages 40-52, Oct. 1992.

[49] D. Garlan and M. Shaw. Introduction to software architecture.
In World Scientific Publishing Company, editor, Advances in Software Engineering and Knowledge Engineering, 1993.

[50] P. Kruchten. The 4+1 view model of architecture. IEEE Softw, 1995.

[51] Nenad Medvidovic, David S. Rosenblum, David F. Redmiles, and Jason E. Robbins. Modeling software architectures in the unified modeling language. In ACM Transactions On Software Engineering and Methodology, 2002.

[52] N. Medvidovic and N R. Taylor. A classification and comparison framework for software architecture description languages. IEEE Transactions on Software Engineering, pages 70–93, 2000.

[53] J. Magee, N. Dulay, S. Eisenbach, and J. Kramer. Specifying distributed software architectures. In Proceedings of the Fifth European Software Engineering Conference, Barcelona, September 1995.

[54] Robert Allen. A Formal Approach to Software Architecture. PhD thesis, Carnegie Mellon University, USA, 1997

[55] D. Garlan, R.T. Monroe, and D. Wile. Acme: Architectural description of component-based systems. In Gary T. Leavens and Murali Sitaraman, editors,

Foundations of Component-Based Systems, Cambridge University Press, pages 47-68, 2000.

[56] J.-M. Jézéquel, S. Gérard, and B. Baudry. Le génie logiciel et l'idm : une approche unificatrice par les modèles. L'ingénierie dirigée par les modèles, Lavoisier, Hermes-science, 2006.

[57] Kassem Saleh; Christo El Morr; Aref Mourtada; Yahya Morad A mobile-agent platform and a game application specifications using MUML The Electronic Library; 2004; 22, 1; Research Library pg. 32

[58] O.Gutknecht, and J. Ferber, Madkit: A Generic Multi-Agent Platform. Autonomous Agents (AGENTS 2000), Barcelona, ACM Press, pp. 78-79, 2000.

[59] J. Sudeikat, L. Braubach, A. Pokahr, and W. Lamersdorf. Validation of BDI Agents. In The Fifth International Workshop on Programming Multi-agent Systems (PROMAS-2006), 2006.

[60] F. Bellifemmine, A. Poggi, and G. Rimassa. JADE - A FIPA compliant agent framework. In 4th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents, 1999.

[61] Schéma directeur des espaces numériques de travail v4, Ministère de l'Éducation nationale, 17 septembre 2012.

[62] Jean-Paul Droz, Créer un Espace numérique de travail en milieu scolaire, Territorial édition, 01/01/2008, ISBN : 978-2-35295-333-3.