MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique
Département de Mathématiques et d'Informatique
Filière Informatique

Titre

Renforcement de la sécurité des systèmes d'information modélisés en UML en utilisant les modèles de contrôle d'accès

Etudiants :
Sakina DJELLOULI

Fatiha OUKACI

Encadrant :

Mr.KHELIFANor Eddine

Deuxième Année Master Ingénierie des Systèmes d'Information
Année Universitaire 2014/ 2015

Dédicace

C'est avec l'immense plaisir et le grand honneur que je dédie ce mémoire :

A mes très chers parents qui ont toujours été là pour moi, et qui m'ont donné un magnifique modèle de labeur et de persévérance. J'espère qu'ils trouveront dans ce travail toute ma reconnaissance et tout mon amour

Pour son précieux soutien, pour sa patience, pour son sourire réconfortant, je remercie, sans doute pas assez, mon chère mari « belkacem »
et je lui dis « je tiens à exprimer toute ma reconnaissance et un immense remerciement à toi pour ta confiance en moi ta encouragement et ta patience avec moi, merci».

À mes chères frères « Mohamed », « Oussama » et à messœurs « Nafissa» «houria» et je leurs dis Je vous aime très fort.

À ma belle famille: ma belle mère que j'aime beaucoup et monbeau père.
À tout membre de ma grande famille: tantes, oncles, cousins et cousines.

À mes enseignants de la filière Informatique et le grand remerciement à mon encadreur Mr. KHELIFA Nor Eddine.

À tous ceux qui ont contribué de près ou de loin à l'édition de ce mémoire.

Sakina

Dédicace

Tout d'abord, je tiens à remercie DIEU le miséricordieux de nous avoir donnés la possibilité de réalisation nos projet, d'arriver à nos souhaits et d'atteindre nos objectifs.

J'aimerai dans ces quelques lignes remercier toutes les personnes qui d'une manière ou d'une autre, ont contribué au bon déroulement de notre travail, tout au niveau humain qu'au niveau scientifique.

Je tiens tout d'abord à remercier notre encadreur Mr : KHELIFA Nor Eddine on a pu bénéficier à la fois de ses compétences scientifiques, et de sa grande disponibilité, tant pour résoudre les difficultés rencontrées lors de nos réalisation, de répandre à nos questions.

J'ajoutai en particulière sa patience et ses encouragements, nousavons permis de travailler dans bonnes conditions.

Jetiens à adresser ma remerciement au membre de jury pour l'intérêt qu'ils ont portés pour notre travail en participant à ce jury en tant qu'examinateurs.

A mes très chers parents qui ont tant prié à ma réussite, De soutien moral. Je les remercie de mes avoir encouragée Et aidé à devenir ce que nous somme. Et A mes frères, ma sœurs, mon Binôme SAKINA, ma grand-parent, et toute la famille OUKACI et Djellouli.

A nos très chers amis qui sont contribués dans notre travail de proche ou de loin plus particulièrement.

fatiha

Résumé

UML est le langage unifié de modélisation qui est le plus utilisé actuellement dans le monde, c'est un moyen qui permet l'analyse et la conception des applications, toute fois ce langage ne permet pas de vérifier la cohérence du modèle fonctionnel avec la politique de contrôle d'accès, ce qui génère des failles dont profitent les attaquants.

Le contrôle d'accès permet de contrôler les accès au système, il est classé en trois classes DAC, MAC et RBAC toute fois notre choix c'est porté le MAC pour son efficacité dans les systèmes à forte composante hiérarchique.

L'utilisation de ce modèle de contrôle d'accès nous a permis de palier les problèmes liés aux contrôle d'accès et cela des les premières phases d'analyse d'une application (première étape de développement), ce qui rend l'application plus fiable et plus résistantes aux menaces des hackers.

Table des matières

Table des matières

Table des matières

Table des matières

Liste des figures

Liste des figures

Liste des figures

Introduction Générale

La sécurité des applications est devenue un enjeu majeur pour la plupart des entreprises à cause du nombre d'attaques de personnes non autorisées toujours croissants qui ne cessent de mettre en danger l'existence même de ces entreprises.

De ce fait, plusieurs solutions ont été proposées qui peuvent être classées en deux grandes catégories : fonctionnelles et non fonctionnelles. La sécurité fonctionnelle considère la sécurité comme un mécanisme de protection externe comme les Firewalls, Antivirus etc…., tandis que la sécurité non fonctionnelle se définit comme un mécanisme de protection interne pour améliorer de sécurité des besoins fonctionnels de l'application.

Les besoin fonctionnels de l'application sont formulés dans le cahier de charge et analysés dans la phase d'analyse de la modélisation de l'application, les besoins non fonctionnels de la sécurité sont représentés par la politique de contrôle d'accès.

Plusieurs modèles de contrôle d'accès ont été proposés, ces modèles se classent en trois grandes catégories: les politiques discrétionnaires (ou DAC pour Discretionary Access Control), les politiques obligatoires (ou MAC pour Mandatory Access Control) et le contrôle d'accès basés sur la notion de rôles (ou RBAC Role-Based Access Control).

UML est le langage unifié de modélisation qui est le plus utilisé actuellement dans le monde, c'est un moyen qui permet l'analyse et la conception des applications, toute fois ce langage ne permet pas de vérifier la cohérence du modèle fonctionnel avec la politique de contrôle d'accès, ce qui génère des failles dont profitent les attaquants.

Pour pallier à ce problème (éviter les failles), nous avons proposé comme thème d'étude de projet le renforcement de la sécurité des systèmes d'information modélisés en UML en utilisant les modèles de contrôle d'accès particulièrement dans la phase d'analyse (les premières phases de developpement)

Notre projet est structuré en trois chapitres :

Chapitre 1. UML et contrôle d'accès

On présente le langage de modélisation unifié UML, ses diagrammes et ses mécanismes d'extension. Ensuite, nous faisons une description détaillée sur la méta-modélisation, plus particulièrement méta-modèle UML, conformément à la définition standardisée par l'OMG.

Ensuite, on définit la sécurité, les politiques de sécurité et les contrôles d'accès ainsi que leurs différents types (DAC, MAC et RBAC). Enfin, nous étudierons plus particulièrement le modèle MAC..

Chapitre 2. Analyse et Conception

Il constitue le cœur de notre travail, nous proposons l'analyse et la conception de notre projet. Nous définissons tout d'abord, le cahier de charge, on passera ensuite à l'analyse et conception à travers les différents diagrammes UML en utilisant la méthode RUP.

Chapitre 3.l' implémentation

Dans ce chapitre nous avons mentionné avec détail les outils que nous avons utilisés pour réaliser notre application, JAVA, XMI, l'API JDOM , et enfin les principales interfaces et fenêtres de l'application.

Chapitre I
UML et Contôle d'accès

I. Introduction

La sécurité d'un système informatique a pour but la protection des ressources (incluant les données et les programmes) contre la révélation, la modification ou la destruction accidentelle ou malintentionnée, tout en garantissant l'accès pour les utilisateurs légitimes.

Pour assurer la sécurité, plusieurs techniques sont employées dont le contrôle d'accès, la cryptographie... Dans le cadre de notre travail, nous allons présenter la modélisation orientée objet en axant notre étude sur UML (Unifiedmodeling langage), qui est devenu le modèle incontournable pour les spécialistes de développement d'applications, et enfin nous étudierons le contrôle d'accès.

II. Définition d'UML

UML (*UnifiedModelingLanguage ou «langage de modélisation unifié»*) est un langage de modélisation graphique à base de pictogrammes, il est apparu dans le monde du génie logiciel, dans le cadre de la «conception orientée objet». Il est couramment utilisé dans les projets logiciels.

UML est utilisé pour spécifié, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet, UML est un langage de modélisation objet qui facilite l'expression et la communication de modèles en fournissant des symboles. UML permet de représenter un système selon différentes vues complémentaires : les diagrammes.

III. Les Diagrammes d'UML

Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle. Chaque type de diagramme UML possède une structure. Les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système (Fig.I.1) **[1]**.

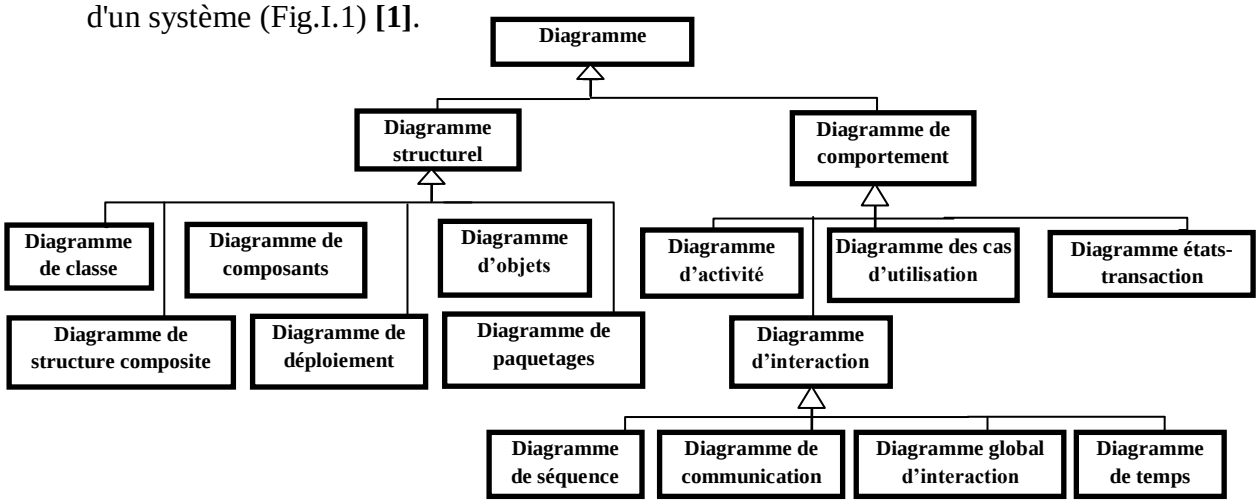


Fig I.1. La hiérarchie des diagrammes UML 2.5.2.

III.1. Diagrammes structurels (statiques)

- ***Diagramme de classes :*** le but d'un diagramme de classes est d'exprimer de manière générale la structure statique d'un système en termes de classes et de relations entreces classes. Une classe a des attributs, des opérations et des relations avec d'autres classes.

- ***Le diagramme d'objet :*** il sert à représenter les instances de classes (objets), Il montre des objets et les liens entre eux.
- ***Diagramme de composants :*** il montre les composants du système du point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques,..). Il permet de mettre en évidence les dépendances entre les composants (*qui utilise quoi*).
- ***Diagramme de déploiement :*** il montre la disposition physique du matériel qui compose le système (ordinateurs, périphériques, réseaux...) et la répartition des composants sur ces matériels. Les ressources matérielles sont représentées sous forme de nœuds, connectés par des supports de communication.
- ***Diagramme des paquetages :*** un paquetage est un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML, il sert à représenter les dépendances entre paquetages.
- ***Diagramme de structure composite :*** est un ensemble d'éléments interconnectés collaborant dans un but commun lors de l'exécution d'une tâche, il est représenté par un ensemble de pièces (rôles) qui sont liés par des connecteurs.

III.2. Diagrammes comportementaux (dynamiques)

- ***Diagramme des cas d'utilisation :*** Il permet à représenter les besoins des utilisateurs par rapport au système. Il montre les relations entre les acteurs et les cas d'utilisation du système.
- ***Diagramme d'activité :*** est une variante des diagrammes d'états-transitions. Il permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation. Dans un diagramme d'activité les états correspondent à l'exécution d'actions ou d'activités et les transitions sont automatiques.
- ***Diagramme états-transitions :*** permet de décrire sous forme de machine à états finis des comportements du système ou de ses composants. Il est composé d'un ensemble d'états, reliés par des arcs orientés qui décrivent les transitions.
- ***Diagramme de séquence :*** Il représente séquentiellement le déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs. Il sert à modéliser les aspects dynamiques des diagrammes des systèmes temps réels et des scénarios complexes. Dans ce type de diagramme, l'accent est mis sur la technologie des envois de messages.
- ***Diagramme de communication :*** c'est une représentation simplifiée d'un diagramme de séquence, en se concentrant sur les échanges de messages entre les objets.
- ***Diagramme global d'interaction :*** il fournit une vue générale des interactions décrites dans le diagramme de séquence et des flots de contrôle décrits dans le diagramme d'activités.
- ***Diagramme de temps :*** il permet de présenter l'interaction entre les objets actifs et leurs changements d'état sur un axe de temps. Il décrit les variations d'une donnée au cours du temps.

IV. Méta-Modèle

Dans le domaine de l'informatique, Le méta-modèle est la définition des constructions et des règles de création des modèles. Donc un méta-modèle est un modèle qui définit le langage d'expression d'un modèle, c.-à-d. le langage de modélisation. Un langage de

modélisation est défini par une syntaxe **abstraite**, syntaxe **concrète** et **sémantique [2]**. La syntaxe *abstraite* définit les concepts de base du langage. La syntaxe *concrète* définit le type de notation qui sera utilisé et donne à chaque concept abstrait une représentation concrète dans cette notation, qui peut être graphique, textuelle ou mixte. Enfin, La *sémantique* définit comment les concepts du langage doivent être interprétés **[3].**

Le méta-modèle qui est une abstraction mettant en évidence les concepts utilisés pour définir le modèle qui représente un phénomène du monde réel, donc un méta-modèle est une « définition formelle » d'un modèle qui aide à le comprendre et qui facilite le raisonnement sur sa structure, sa sémantique et son usage.

Par exemple, XML (*Extensible Markup Langage « langage de balisage extensible »)* pour lequel la DTD joue le rôle de méta-modèle pour des documents XML pouvant alors être considérés comme des modèles.

V. Méta-modélisation d'UML

Le méta-modèle d'UML définit la structure que doit respecter tout modèle UML. Ainsi, les différents concepts du langage UML ont été eux-mêmes modélisés avec UML.

L'approche de méta-modélisation adoptée par l'OMG est définie par une architecture en quatre couches (voir la Fig.I.2).

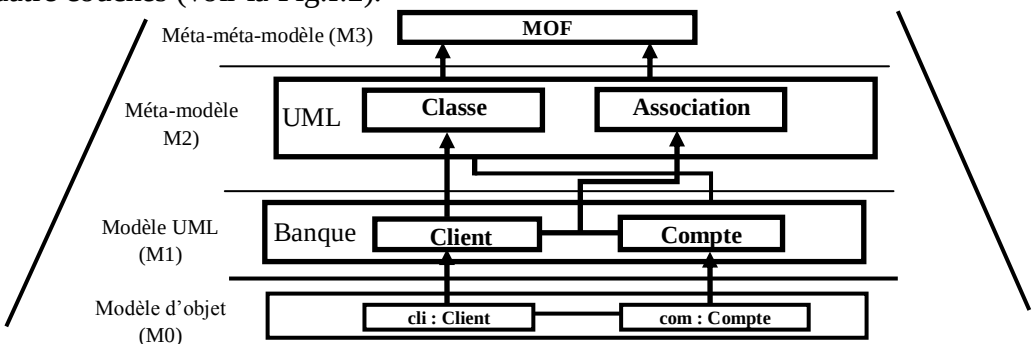


Fig I.2.L'architecture l'UML à quatre niveaux de l'OMG.

- **Niveauméta-méta-modèle (M3) :** M3 est composé d'une unique entité qui s'appelle le MOF (*Meta Object Facility*). MOF définit un langage abstrait permettant de décrire la structure des méta-modèles, d'étendre ou de modifier les méta-modèles existants **[3]**.
- **Niveau méta-modèle (M2) :** le méta-modèle d'UML se situe à ce niveau qui est décrit dans le standard UML; il définit la structure interne des modèles UML et il est spécifié en utilisant le MOF, c.à.d. les concepts du méta-modèle d'UML sont des instances des concepts de MOF. La figure I.2 montre deux méta-classes du méta-modèle UML: *Classe* et *Association*.
- **Niveau modèle (M1) :** M1 correspond au niveau des modèles UML des utilisateurs. Les concepts d'un modèle UML sont des instances des concepts du méta-modèle UML. La figure I.2 montre un extrait de diagramme de classes pour une application bancaire contenant deux classes *Compte* et *Client* liées par une association UML. Les deux classes sont des instances de la méta-classe Classe et le lien est une instance de la méta-classe Association du méta-modèle UML.
- **Niveau objets (M0) :** M0 correspond au niveau des objets réels, il est composé des informations que l'on souhaite modéliser. Ce niveau est souvent considéré comme

étant le monde réel **[6].** Il s'agit ici de deux objets *cli* et *com* des instances des deux classes *Client* et *Compte* respectivement.

Le méta-modèle d'UML est décrit en utilisant une partie de la notation d'UML lui-même. Les concepts suivants sont utilisés :

– Les classes d'UML, pour décrire les *méta-classes*.

– Les attributs, pour décrire les propriétés attachées à une méta-classe.

– Les associations, pour décrire des liens entre les méta-classes.

– Les paquetages (packages), pour regrouper les méta-classes par domaine **[4]**.

V.1. Le méta-modèle de diagramme de classes

Le méta-modèle de diagramme de classes (Fig.I.3) contient des packages. Un *package* à un nom et contient des classes. Un package peut *importer* un autre package. Une *classe* a un nom et peut *contenir* des attributs et des associations. Une classe peut aussi *hériter* d'une autre classe. Un *attribut*àun nom et une visibilité (public, privé ou protégé), Une *association*peut être agrégation ou composition. Un attribut et Association sontun type qui peut être un type de base (string, integer, boolean).

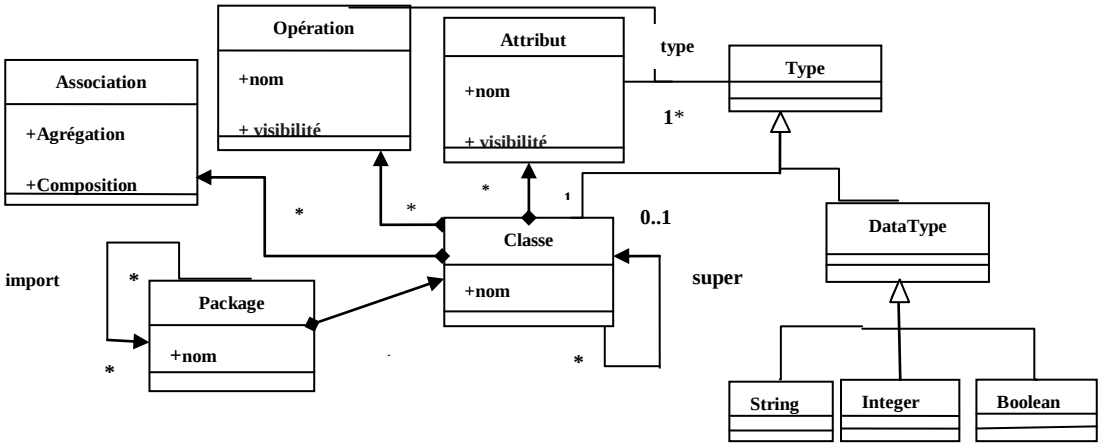


Fig I.3. Le méta-modèle de diagramme de classes UML.

V.2. Méta-modèle de diagramme de cas d'utilisation

Le méta-modèle de diagramme de cas d'utilisation (Fig.I.4) contient une classe acteurs, classe système et classe cas d'utilisation. Un acteur a un nom et est *relié* aux cas d'utilisation. Un acteur peut *hériter* d'un autre acteur. Un cas d'utilisation a un intitulé et peut *hériter, étendre ou inclure*un autre cas d'utilisation. Le système a lui aussi un nom, et il inclut tous les cas d'utilisation **[5]**.

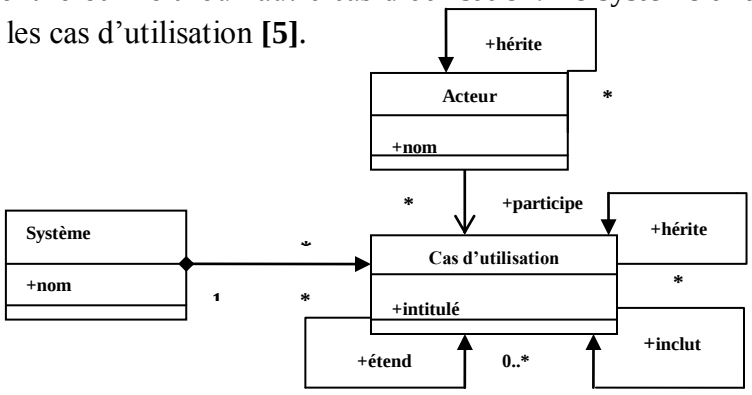


Fig I.4. Méta-modèle des diagrammes de cas d'utilisation.

- **Exemple de Méta-modèle de diagramme de cas d'utilisation**

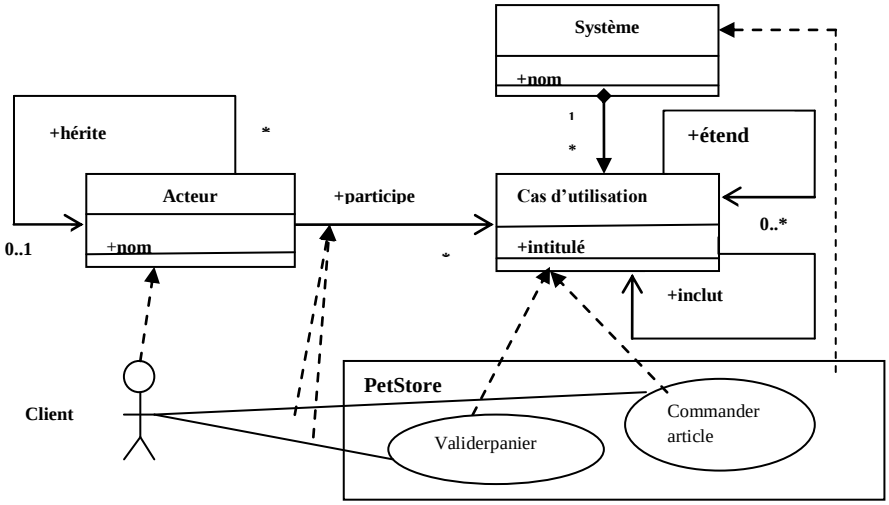


Fig I.5. Relations entre un diagramme de cas d'utilisation et son méta-modèle.

V.3. Méta-modèle de diagramme de séquence

Le méta-modèle des diagrammes de séquence est composé de Séquence-Diagramme Cette classe a pour fonction d'associer les messages aux objets qui communiquent, et ainsi construire le diagramme de séquence final,Objet représente les objets du diagramme de séquence. Ces objets envoient et reçoivent des messages dans un ordre chronologique. Chaque objet est désigné par son nomet Message représente les messages échangés entre les objets qui lui correspondent. Chaque message est étiqueté par le nom de l'opération, ils peuvent être synchrones ou asynchrones [7].

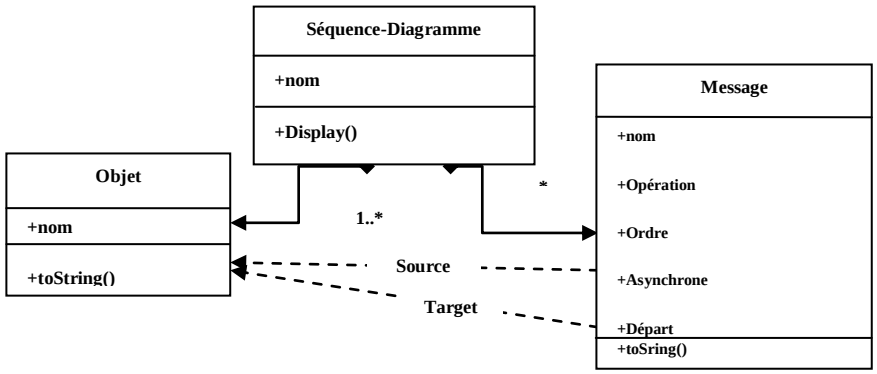


Fig I.6. Méta-modèle des diagrammes de séquence.

VI. Mécanismes d'extension d'UML

UML fournit des mécanismes d'extension du langage : stéréotypes, étiquettes (valeur marquée), notes, contraintes et Profils.

VI.1. Stéréotypes

Un *stéréotype* constitue un moyen de classer les éléments de la modélisation. Il introduit une nouvelle classe dans le méta-modèle par dérivation d'une classe existante. Le nom du stéréotype est placé entre guillemets **(<<>>)** avant le nom de l'élément auquel il s'applique.

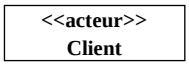


Fig I.7. Représentation d'une classe stéréotypée.

VI.2. Etiquette (valeur marquée)

Une valeur marquée est une paire **(nom, valeur)** qui ajoute une nouvelle propriété à un élément de modélisation. La spécification d'une valeur marquée prend la forme : **nom = valeur**. Une valeur marquée est indiquée entre accolades.

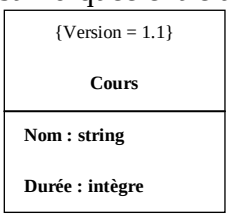


Fig I.8. Exemple d'une classe étiquetée

VI.3. Note

Une note est un symbole graphique qui contient des informations. L'utilisation d'une note permet de présenter des commentaires, des contraintes ou des valeurs marquées. Elle est représentée par un *rectangle écorné lié* par un ou plusieurs traits pointillés aux éléments qu'elle commente.

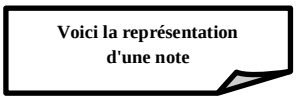


Fig I.9. Représentation d'une note.

VI.4. Contrainte

Une contrainte est une note ayant une valeur sémantique particulière pour un élément de la modélisation. Pour exprimer des contraintes de manière formelle, le langage de contraintes OCL (*Object ConstraintLanguage*) peut être utilisé. Chaque contrainte est indiquée entre accolades {}. Une contrainte peut être associée à plusieurs éléments par des relations de dépendance **[8]**.

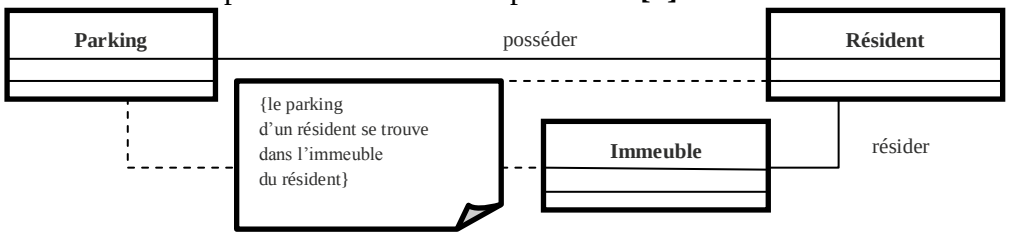


Fig I.10. Exemple d'utilisation d'une contrainte OCL.

VI.5. Profils

Le mécanisme de profil permet d'étendre ou de restreindre le méta-modèle d'UML de manière à l'adapter à un usage spécifique, les nouveaux éléments peuvent être des stéréotypes, des valeurs étiquetées ou des contraintes spécifiées pour un projet donné **[6]**.

La Figure ci-dessous montre un exemple du profil UML appelé Exemple regroupant les deux stéréotypes Horloge et Créateur. Les profils dans UML sont notés comme des paquetages avec le stéréotype <<profile>>. Le paquetage UserExemple est un exemple de modèle utilisateur basé sur le profil Exemple. La classe Chronomètre est définie avec les deux stéréotypes de profils et les valeurs marquées associées aux stéréotypes sont définis comme des notes UML (Fig.I.11).

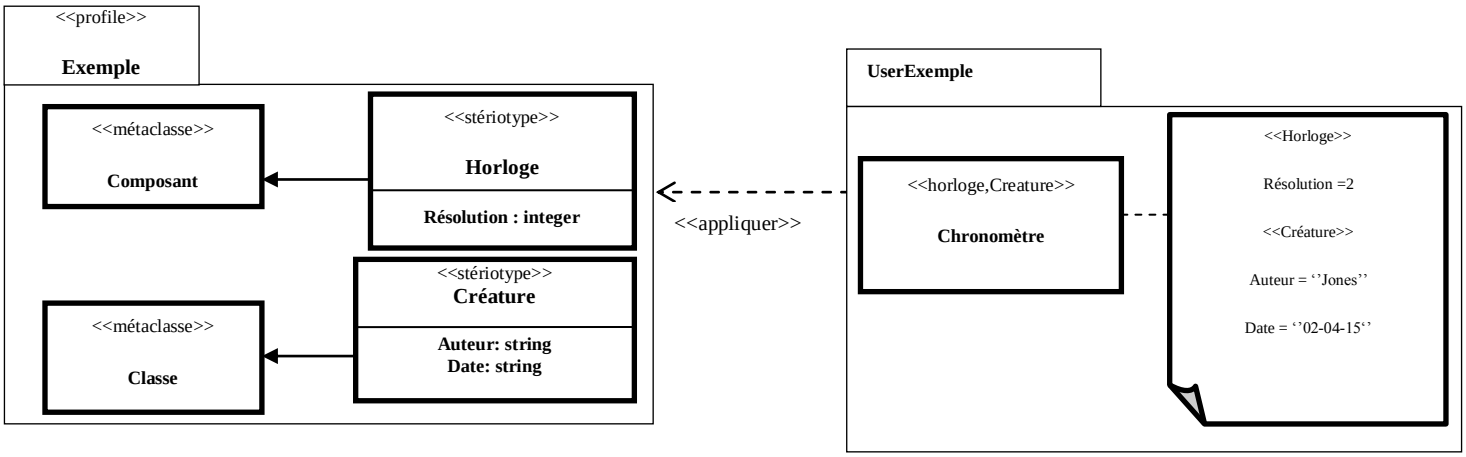


Fig I.11. Exemple de spécification d'un profil UML et son utilisation.

VII. La sécurité des systèmes d'information

VII.1. Définition de la sécurité

Dans le domaine de l'informatique, le mot "sécurité" peut avoir plusieurs définitions. La première correspond à la sécurité-innocuité (en anglais *safety*) et concerne la prévention de catastrophes : dans ce sens, un système informatique aura une sécurité satisfaisante si aucune de ses défaillances éventuelles ne peut provoquer de dégâts importants.

La seconde définition du terme de sécurité correspond au mot anglais "*security*" et concerne la capacité du système informatique à résister à des agressions externes physiques (incendie, inondation, bombes, etc.) ou logiques (erreurs de saisie, intrusions, piratages, etc.).

Ce sont les spécialistes de l'audit de sécurité qui définissent la sécurité informatique pour chaque entreprise donnée selon leurs évaluations des risques **[9]**.

La troisième celle de l'ITSEC **[10]**, qui considère la sécurité comme la combinaison de trois propriétés : la *confidentialité, l'intégrité* et la *disponibilité* de l'information.

- **La confidentialité** permet d'empêcher la divulgation non-autorisée de données ;
- **L'intégrité** permet d'empêcher la modification non-autorisée de données;
- **La disponibilité** est la propriété d'une information d'être accessible lorsqu'un utilisateur autorisé en a besoin.

En plus de cette combinaison des trois propriétés, la sécurité compte de nouveau autres propriétés, que nous détaillons ci-dessous :

- **La non répudiation**: permet de garantir qu'une transaction ne peut être niée;
- **L'authenticité** : permet d'empêcher l'utilisation non-autorisée de ressources;
- **La responsabilité**: regroupe la disponibilité et l'intégrité de l'identité de la personne qui a effectué une opération **[11]**.

VII.2. La politique de sécurité

La politique de sécurité est l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique **[10]**. Elle se développe selon trois axes: physique, administratif et logique.

Le premier précise l'environnement physique du système à protéger (catastrophes). Le deuxième décrit les procédures organisationnelles (répartition des tâches, séparation des

pouvoirs). Le troisième a trait aux contrôles d'accès logiques (qui, quoi, quand, pourquoi, comment) et s'intéresse aux fonctions d'identification, d'authentification et d'autorisation mises en œuvre par le système informatique**[11].** Dans ce chapitre, on s'intéresse à la sécurité logique, et plus particulièrement aux contrôles d'accès.

VII.3. Le contrôle d'accès

Il est utile de rappeler que l'une des première mesures en sécurité informatique consiste à contrôler les différents accès possible à un système d'information et à autoriser ou non un certain nombre d'actions en fonction de l'utilisateur. Il s'agit de limiter l'accès aux ressources du système d'information uniquement aux utilisateurs, programmes, procédés ou systèmes autorisés.

Le contrôle d'accès est défini comme n'importe quel mécanisme par lequel un système autorise ou interdit le droit à des entités actives (sujets) d'accéder à des entités passives (objets), ou d'effectuer des opérations.

VII.3.1. Les concepts fondamentaux d'une politique de contrôle d'accès

- **Sujet :** entité active qui accède aux données du système. Le sujet peut être un utilisateur, une application, une adresse IP…
- **Objet :** entité passive qui représente les données à protéger. L'objet peut être, par exemple, un fichier, une table relationnelle, une classe …
- **Action :** représente l'action à traiter par le sujet sur l'objet. L'action peut être : lire, écrire, exécuter …**[12].**

VII.3.2. Les modèles de contrôle d'accès

On distingue principalement trois grandes catégories de modèles de contrôle d'accès : les modèles de contrôle d'accès discrétionnaires DAC (Discretionary Access Control) et le modèle de contrôle d'accès obligatoires MAC (Mandatory Access Control). Afin de mieux s'adapter à des organisations particulières, d'autres modèles ont été définis, en particulier, le modèle de contrôle d'accès basé sur la notion de rôle RBAC (RoleBased Access Control) **[9]**.

VIII. Modèle de contrôle d'accès discrétionnaire (DAC)

Le modèle DAC (Discretionary Access Control) est un modèle de contrôle d'accès défini comme un «moyen de restreindre l'accès aux objets en fonction du sujet (ou du groupe) auxquels ils appartiennent». L'aspect discrétionnaire dans le sens où le sujet est capable de transférer les permissions d'accès à d'autres sujets («à la discrétion du propriétaire»)**[13]**.

IX. Contrôle d'accès basé sur les rôles (RBAC)

Le RBAC (Role-Based Access Control) est un modèle de contrôle d'accès à un système d'information dans lequel chaque décision d'accès est basée sur le rôle auquel l'utilisateur est attaché. Un rôle découle généralement de la structure d'une entreprise. Les utilisateurs exerçant des fonctions similaires peuvent être regroupés sous le même rôle. Un rôle, déterminé par une autorité centrale, associe à un sujet des autorisations d'accès sur un ensemble objets **[18]**.

X. Contrôle d'accès obligatoire (MAC)

L'objectif du contrôle d'accès mandataire (MAC) est d'imposer une politique non modifiable par les utilisateurs finaux, Pour contrôler les accès entre sujets et objets.MAC est utilisé principalement dans les environnements militaires à cause de son contrôle centralisé et il permet à l'administrateur du système de définir des privilèges pour protéger la confidentialité et l'intégrité des ressources dans le système.

Dans ce modèle un *niveau de sécurité* est attribué à chaque objet et sujet. Le niveau de sécurité d'un sujet est appelé *Habilitation* alors que le niveau de sécurité d'un objet est appelé *Classification*. Des exemples typiques de ces niveaux sont : « *Public* », « *Confidentiel* », « *Secret* » ou « *Top Secret* » (Fig.I.12).

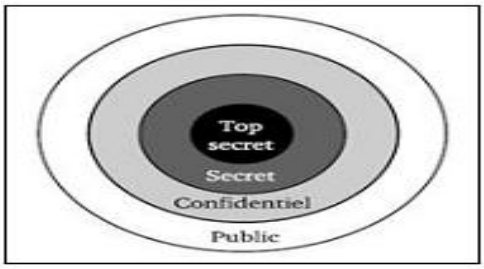


Fig I.12. Les niveaux de sensibilité dans le modèle MAC.

Les politiques obligatoires les plus fréquemment utilisées sont les politiques multi-niveaux(MLS). Ces politiques reposent sur des classes de sécurité affectées aux informations et des niveaux des habilitations affectées aux utilisateurs**[16]**.

X.1. Politique du modèle de Bell-LaPadula

Ce modèle a été proposé par David Bell et Leonard Lapadula. La politique *multi-niveaux* de BLP est une politique obligatoire, développée pour le *DoD*(*Department of Defense*) des Etats-Unis, concerne la confidentialité des données du monde militaire. Les permissions d'accès sont définies à travers une matrice d'accès et un ensemble de niveaux de sécurité. Cette politique considère seulement les flux d'informations qui se produisent quand un sujet observe ou modifie un objet.

Le modèle associé à la politique de BLP est fondé sur la notion de treillis. Il s'appuie sur l'association de différents niveaux aux sujets (niveaux d'habilitation) et aux objets (niveaux de classification). Chaque niveau $n = (cl, C)$ est caractérisé par ses deux attributs :

C : représente un ensemble de catégories, par exemple {nucléaire, Chimique}.

cl: représente une classification: Top Secret (TS), Secret (S), Confidentiel.

Les niveaux constituent un *treillis* partiellement ordonné par une relation de dominance notée "$\leq$ " et définie par :

si$n = (cl, C)$ et $n' = (cl', C')$; $n \leq n'$ (n' domine n) si et seulement si $cl \leq cl'$ et $C \subseteq C'$

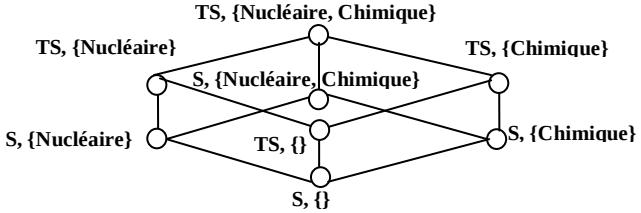


Fig I.13. Présente un exemple de treillis de sécurité.

Ce modèle est défini par deux propriétés : Propriété simple et Propriété étoile**[16]**.

Propriété simple «***No read up***»**:** un sujet $s_i$ne peut lire un objet $o_j$que si son habilitation h (s_i) domine la classification c (o_j) de l'objet : (s_i, o_j, *lire*) =>$h(s_i) \geq c$ (o_j).La figure ci-dessousprésente des exemples d'application :

- Des lignes qui représentent la séparation entre les niveaux de sécurité.
- Des cercles qui représentent les sujets.
- Des carrés qui représentent les objets.
- Des flèches allant des objets vers les sujets qui représentent les actions de lecture.
- Des flèches allant des sujets vers les objets qui représentent les actions d'écriture.
- Une flèche barrée signifie que l'opération correspondante est interdite.

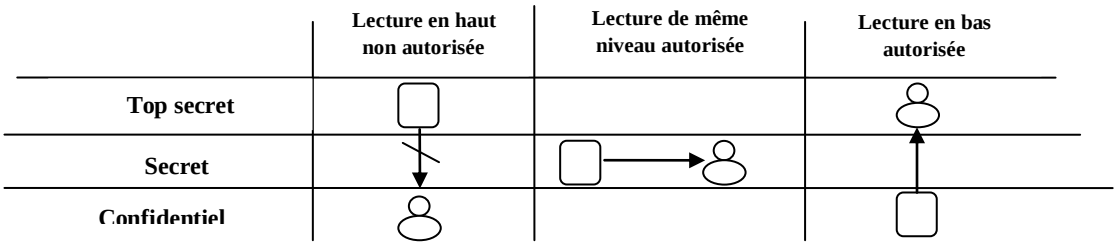


Fig I.14. La propriété simple de BLP

Propriété étoile «No write down»: un sujet ne peut lire un objet $o_j$et en écrire un autre $o_k$que si la classification d'$o_k$domine celle d'o_j:(s_i, o_j, *lire*) $\wedge$ (s_i, o_k, *écrire*) =>$c(o_k) \geq c(o_j)$.

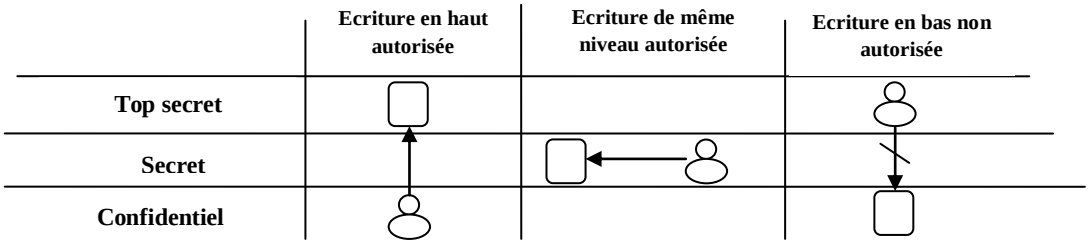


Fig I.15. La propriété étoile de BLP

X.2. Politique d'intégrité de Biba

Ken Biba propose une politique duale de celle de BLP pour assurer l'intégrité. A chaque sujet et objet est associé un niveau d'intégrité qui correspond respectivement au "pouvoir d'accès" et au niveau d'intégrité (du sujet qui l'a créé) [16].

Le modèle Biba est défini par deux propriétés:

La propriété simple «***No read down***»**:** il suffit de ne pas lire vers le bas. Cette loi interdit un sujet d'avoir un accès en lecture à un objet qui a une classification inférieure que de l'habilitation du même sujet: $Lire \in M(s,o) \Rightarrow f(o) \geq f(s)$

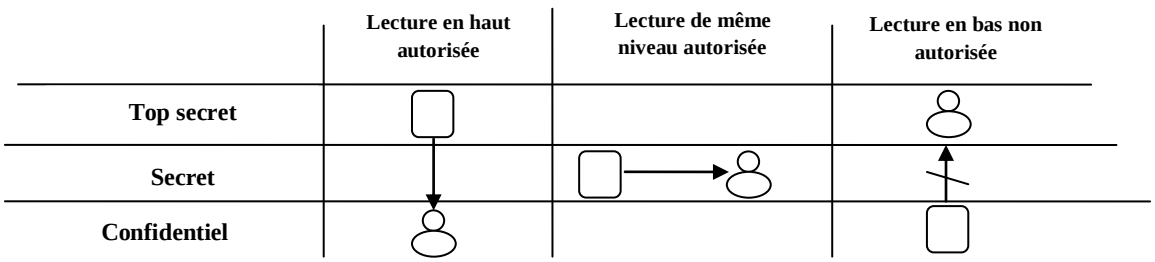


Fig I.16. La propriété simple de Biba

La propriété étoile «***No write up***»**:** cette loi interdit un sujet d'avoir une écriture accès à un objet qui a un classement plus élevé que de l'habilitation du même sujet: *écrire* $\in M(s,o) \Rightarrow f(s) \geq f(o)$

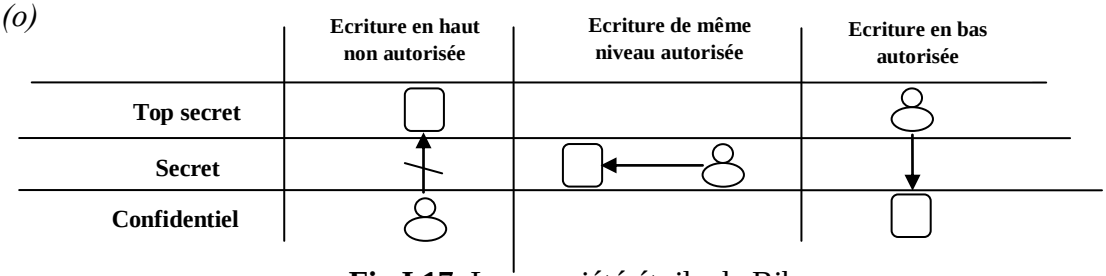


Fig I.17. La propriété étoile de Biba

XI. Présentation de l'application

- **Le but de notre travail**

UML permet de modéliser le cahier de charge fonctionnel de l'application, en utilisant ces différents diagrammes qui représentent les différentes vues sur le système, ces diagrammes sont modélisés avec le méta modèle UML,toute fois UML n'est pas une méthode, pour cela plusieurs démarche ont été créé parmi eux RUP (RationnalUnifiedProcess), MDA (model Driven Architecture) et UP7 (UnifiedProcess 7).

Tous ces démarches utilisent trois diagrammes comme moyen pour analyser l'application et cela des le premier pas de modélisation (cas d'utilisation, séquence, classe).

Cette étude nous a permis de concrétiser un outil de travail capable de :

- introduire les éléments du modèle d'application, on a choisi 3 diagramme du modèle UML, qui sont le diagramme de cas d'utilisation de séquence et de classe.
- l'intégrer le MAC dans les diagrammes UML et ainsi faire des contrôle et des testes.

- **Pourquoi notre travail est-il axé sur les diagrammes de cas d'utilisation, de séquence et de classe ?**
  - Ils sont considérés comme les premiers diagrammes dans la partie analyse de plusieurs processus de développement de logiciel tels que RUP et MDA.
  - Ils permettent de bien modéliser les applications, car ils présentent les trois principales vues proposées par UML : la vue statique, la vue fonctionnelle et la vue dynamique.
- **Pourquoi avons-nous choisi le MAC ?**

Le MAC a été défini pour répondre aux besoins de contrôle d'accès dans des systèmes multi-niveaux, ces points forts résident essentiellement dans :

- La simplicité de mise en œuvre car il correspond bien aux modes de fonctionnements utilisés dans des organisations à forte composante hiérarchique
- La possibilité de définir une politique et des objectifs de sécurité d'une manière précise et de les vérifier.

XI. Conclusion

Dans ce chapitre on a traité deux grande parties, dans la première partie nous avons présenté succinctement le langage de modélisation unifié UML, son évolution, ses diagrammes et ses mécanismes d'extension. Ensuite, nous avons donné une description détaillée sur la méta-modèle UML, qui est adoptée par l'OMG est connue comme une hiérarchie à quatre niveaux.

Dans la deuxième partie, nous avons données une définition détaillée sur la sécurité, ainsi les techniques et mécanismes pour sécuriser un système, ensuite nous nous sommes étendues sur le contrôle d'accès en essayant de décrire les différentes variantes des modèles de contrôle d'accès les plus connus dans la littérature (DAC, MAC, RBAC) rappelant ainsi les caractéristiques de ces modèles, qui sont comme nous l'avons montré indépendants des modèles de données.

Enfin, nous avons présenté notre but du projet et les argumentations sur le choix des diagrammes et du modèle de contrôle d'accès.

Chapitre II
Analyse et Concption

I. Introduction

Dans ce chapitre nous présentons la partie analyse et conception du projet. Ce projet consiste à créer un environnement de développement d'applications sécurisé (contrôle d'accès) c'est-à-dire intégrer le contrôle d'accès dès la spécification des diagrammes de cas d'utilisation, de séquence et classe.

Ainsi notre projet consiste à élaborer une application qui permet d'analyser les besoins de contrôle d'accès à partir des spécifications des besoins fonctionnels.

II. La description du processus

Tout au long de modélisation de cette application, nous avons eu besoin d'utiliser seulement cinq diagrammes parmi les treize proposés par le langage UML. Ce choix nous a permis de bien comprendre le fonctionnement de ces cinq diagrammes et de maitriser leur usage au sien de notre modèle UML.

- Les diagrammes de cas d'utilisation représentant l'aspect fonctionnel.
- Les diagrammes de classes et composant représentant l'aspect statique.
- Les diagrammes de séquence et les diagrammes d'activités représentant l'aspect dynamique.

Lors de la conception, nous avons entrepris une démarche basée sur le processus RUP (Rational UnifiedProcess). Dans ce qui suit, nous allons détailler les différents workflows dont nous nous sommes servis :

L'analyse des besoins : nous l'avons définie deux étapes :

- La collecte des besoins
- La spécification des cas d'utilisation

L'analyse : nous l'avons définie en trois étapes :

- La spécification des diagrammes de classe d'analyse
- La spécification des diagrammes de séquence.
- La spécification des diagrammes d'activités.

L'implémentation : dans cette phase apparier des diagrammes générer nous allons générer du code adéquat

Cette phase sera décrite dans le chapitre suivant.

III. Cahier charge fonctionnel

III.1. Cahier charge de l'application

- L'utilisateur introduit le modèle de son application :
  - L'utilisateur introduit les différents éléments des diagrammes de cas d'utilisation.
  - L'utilisateur introduit les différents éléments des diagrammes de séquences d'analyse.

- L'utilisateur introduit les différents éléments des diagrammes de classe.
- L'utilisateur introduit la politique de contrôle d'accès selon le modèle MAC.
- Le système analyse la cohérence entre le modèle de l'application et la politique de contrôle d'accès

IV. Analyse

IV.1. Vue globale

Notre projet est divisé en deux grandes parties schématisées comme la montre la figure ci-dessous.

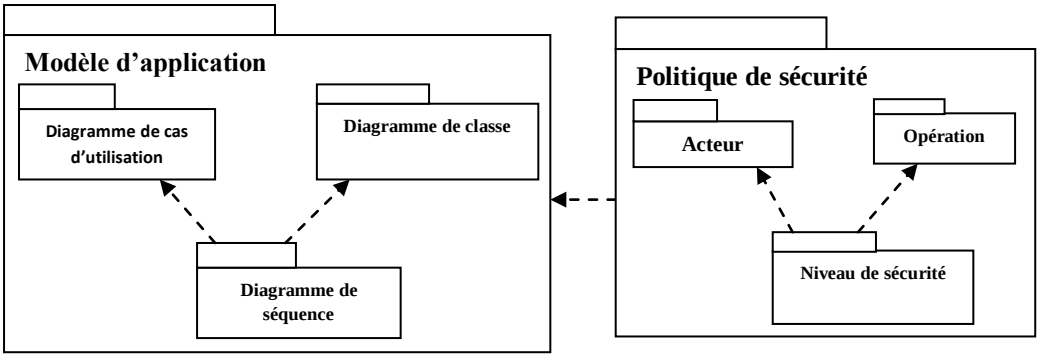


Fig II.18. Diagramme de package de l'application UML_MAC.

1. **Modèle d'application :** dans ce package on identifie les éléments du diagramme de cas d'utilisation et leur scénarios sous forme de diagrammes de séquence ainsi que le diagramme de classe d'analyse.
2. **Politique de sécurité :** dans ce package on extrait les acteurs et les opérations du modèle d'application (cas d'utilisation, classe), l'utilisateur attribut ensuite des niveaux de sécurité (TopSecret, Secret…) pour chaque acteurs et opérations.

IV.2. La spécification des cas d'utilisation

IV.2.1. Diagramme de cas d'utilisation d'application

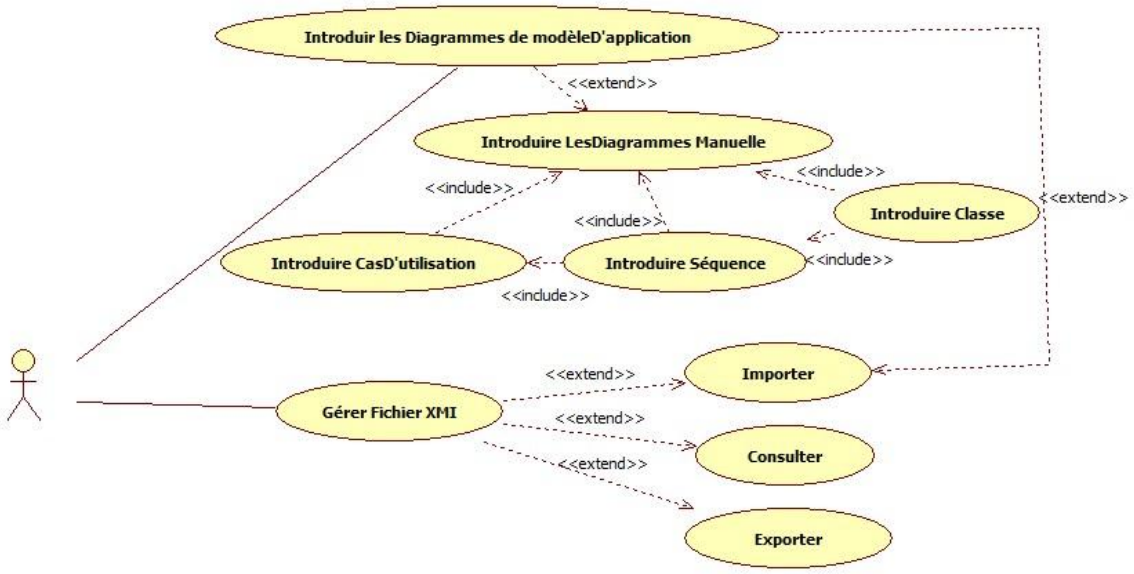


Fig.II.19. Vue de haut niveau des cas d'utilisation de l'application Modèle d'application.

- **Diagramme de cas d'utilisation du diagramme de cas d'utilisation**

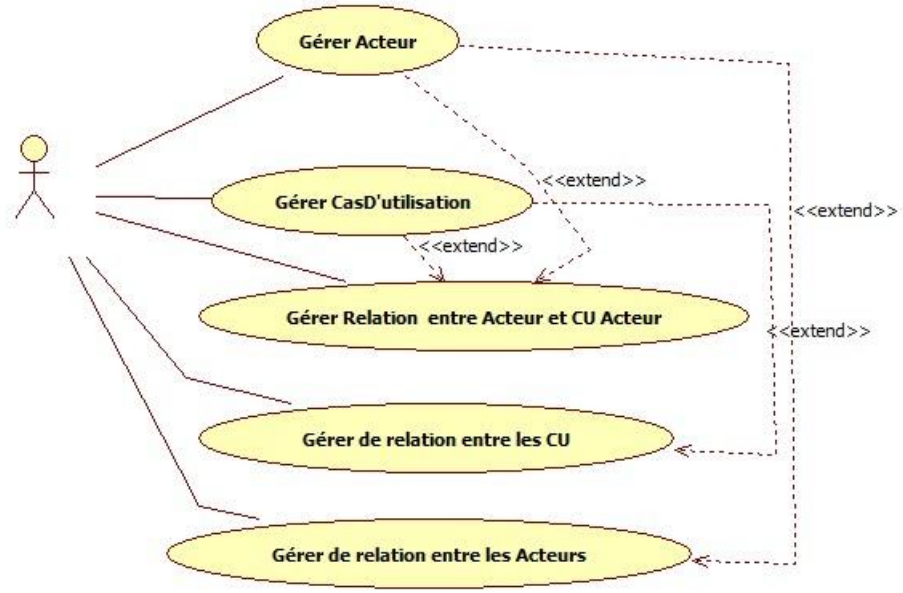


Fig.II.20. Diagramme de cas d'utilisation de diagramme de cas d'utilisation.

- **Description textuelle**

Titre : Gérer Acteur.

Résumé : ce cas d'utilisation permet d'ajouter, modifier et supprimer les acteurs.

Acteur : utilisateur. **Responsable :**DjellouliSakina/Oukaci Fatiha

Pré condition : l'existence les diagrammes sur papier ou système modélisé.

Scénario nominal : Ajouter un Acteur.

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3-Choisir page Acteur.

4-choisir la page Propriété.

5-Entrer le nom d'Acteur.

6-Ajouter Acteur.

7-Vérifier l'existence.

8-Afficher l'Acteur.

Scénario alternatif : Modifier et Supprimer un Acteur.

- **Modifier un Acteur :**

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3-Afficher les listes d'Acteur.

4-Sélectionner un Acteur.

5-Sélectionner la page Acteur.

6-Entrer nouvel nom d'Acteur.

7-Vérifier l'existence.

8-Modifier Acteur.

- **Supprimer un Acteur :**

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3-Afficher les listes d'acteur.

4-Sélectionner un acteur.

5-Sélectionner la page Acteur.

6-Supprimer Acteur.

7-Vérifier l'existence.

Post condition : ensemble des Acteurs correcte.

Titre : Gérer Cas d'Utilisation.

Résumé : ce cas d'utilisation permet d'ajouter, modifier et supprimer les cas d'utilisation.

Acteur : utilisateur. **Responsable :**DjellouliSakina/Oukaci Fatiha

Pré condition : l'existence les diagrammes sur papier ou système modélisé.

Scénario nominal : Ajouter un Cas d'Utilisation.

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de cas d'utilisation.

3-Choisir la page Cas d'Utilisation.

4-Entrer le nom de Cas d'Utilisation.

5-Ajouter Cas d'Utilisation.

6-Vérifier l'existence.

7-Afficher le Cas d'Utilisation.

Scénario alternatif : Modifier et Supprimer un Cas d'Utilisation.

- **Modifier un Cas d'Utilisation :**

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3-Afficher les listes de Cas d'Utilisation.

4-Sélectionner un Cas d'Utilisation.

5-Sélectionner la page Cas d'Utilisation.

4-Entrer nouvel nom de Cas d'Utilisation.

5-Modifier Cas d'Utilisation.

6-Vérifier l'existence.

- **Supprimer un Cas d'Utilisation :**

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3-Afficher les listes de Cas d'Utilisation.

4-Sélectionner un Cas d'Utilisation.

5-Sélectionner la page Cas d'Utilisation.

6-Supprimer le Cas d'Utilisation.

7-Vérifier l'existence.

Post condition : ensemble des Cas d'Utilisation correcte.

Titre : Gérer relation entre Acteur et Cas d'Utilisation.

Résumé : ce cas d'utilisation permet d'ajouter, modifier et supprimer les relations entre Acteur et Cas d'Utilisation.

Acteur : utilisateur. **Responsable :**DjellouliSakina/Oukaci Fatiha

Pré condition : l'existence les diagrammes sur papier ou système modélisé.

Scénario nominal : Ajouter une relation entre Cas d'Utilisation et Acteur.

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3-Choisir la page AssociationActeur/Casd'Utilisation.

4-Sélectionner Acteur.

5-Sélectionner Cas d'Utilisation.

6-Ajouter une Association.

7-Vérifier l'existence.

8-Afficher l'Association.

Scénario nominal : Modifier et Supprimer une relation entre Cas d'Utilisation et Acteur.

- **Modifier une relation entre Cas d'Utilisation et Acteur :**

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3-Afficher les listes AssociationActeur/Casd'Utilisation.

4-Sélectionner une Association.

5-choisir la page AssociationActeur/Cas d'Utilisation.

6-Modifier Extrémité Acteur.

7-Modifier Extrémité Cas d'Utilisation.

8-Modifier Association.

9-Vérifier l'existence.

- **Supprimer une relation entre Cas d'Utilisation et Acteur :**

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3-Afficher les listes Association Acteur/Casd'Utilisation.

4-Sélectionner une Association.

5-Choisir la page AssociationActeur/Casd'Utilisation.

5-Supprimer Association.

6-Vérifier l'existence.

Titre : Gérer relation entre Acteur.

Résumé : ce cas d'utilisation permet d'Ajouter, Modifier et Supprimer les relations entre Acteur.

Acteur : utilisateur. **Responsable :**DjellouliSakina/Oukaci Fatiha

Pré condition : l'existence les diagrammes sur papier ou système modélisé.

Scénario nominal : Ajouter une relation entre l'Acteur.

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3-Choisir la page Généralisation.

4-Sélectionner l'Acteur fils.

5-Sélectionner l'Acteur Parent.

6-Ajouter une Généralisation.

7-Vérifier l'existence.

8-Afficher Généralisation.

Scénario nominal : Modifier et Supprimer une relation entre l'Acteur.

- **Modifier une relation entre l'Acteur :**

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3-Afficher les listes de Généralisation.

4-Sélectionner un Généralisation.

5-Choisir la page Généralisation.

6-Modifier l'Acteur fils.

7-Modifier l'Acteur Parent.

8-Modifier Généralisation.

9-Vérifier l'existence.

- **Supprimer une relation entre l'Acteur :**

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3-Afficher les listes de Généralisation.

3-Sélectionner un Généralisation.

6-Supprimer Généralisation.

7-Vérifier l'existence.

- **Diagramme de cas d'utilisation de diagramme de séquence**

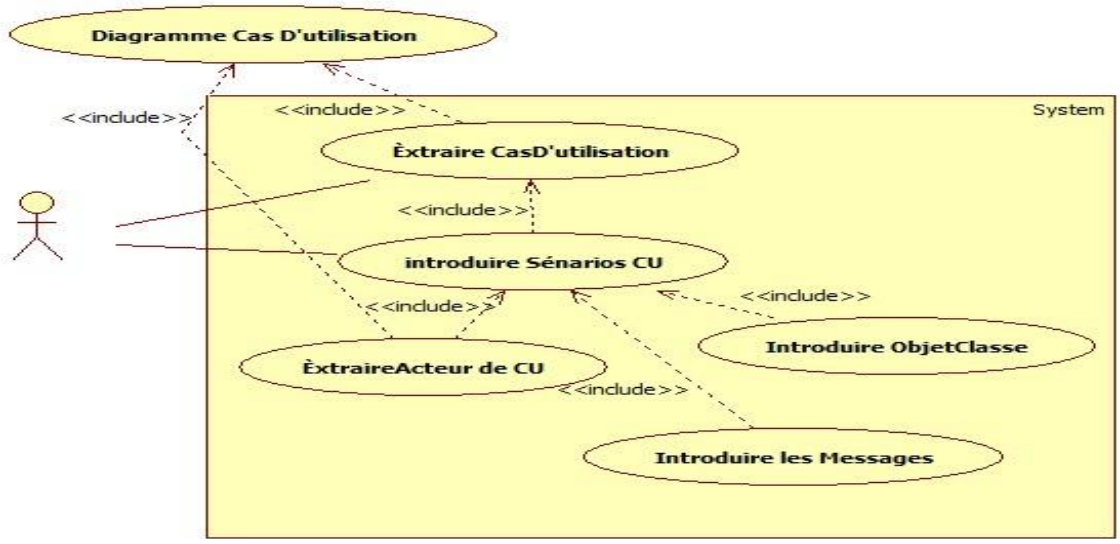


Fig.II.21. Vue haut niveau des cas d'utilisation de diagramme de séquence.

- **Description textuelle**

Titre : Introduire ObjetClasse.

Résumé : ce cas d'utilisation permet d'ajouter, modifier et supprimer les ObjetClasse.

Acteur : utilisateur. **Responsable :**DjellouliSakina/Oukaci Fatiha

Pré condition : l'existence les diagrammes sur papier ou système modélisé.

Scénario nominal : Ajouter un ObjetClasse.

1- Accéder à l'application UML_MAC.

2-Afficher la page diagramme de Cas d'Utilisation.

3- Choisir la page Diagramme de Séquence.

4-Extraire les CU.

5-Sélectionner un CU.

6-Afficher les Acteurs de CU à sélectionner.

7-Sélectionner un Acteur.

8-choisir la page ScénarioActeur.

9-Choisir la page Interaction.

10-Entrer le nom ObjetClasse.

11-AjouterObjetClasse.

12-Vérifier l'existence.

13-Afficher l'ObjetClasse.

Scénario alternatif : Modifier et Supprimer un ObjetClasse.

Titre : Introduire les Messages.

Résumé : ce cas d'utilisation permet d'Ajouter, Modifier et Supprimer les Message.

Acteur : utilisateur. **Responsable :**DjellouliSakina/Oukaci Fatiha

Pré condition : l'existence les diagrammes sur papier ou système modélisé.

Scénario nominal : Ajouter un Message.

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3- Choisir la page Diagramme de Séquence.

4-Extraire les CU.

5-Sélectionner un CU.

6-Choisir la page Acteurs.

7-Afficher les Acteur de CU à sélectionner.

8-Sélectionner un Acteur.

9-Choisir la page ScénarioActeur.

10-Choisir la page Interaction.

11-Entrer le nom de l'ObjetClasse.

12-Ajouter ObjetClasse.

13-Vérifier l'existence.

14-Afficher l'ObjetClasse.

15-Sélectionner l'ObjetClasse.

16-Choisir la page Message d'interaction.

17-Entrer le nom de Message.

18-Ajouter Message.

19-Vérifier l'existence.

13-Afficher le Message.

- **Diagramme de cas d'utilisation de diagramme de classe**

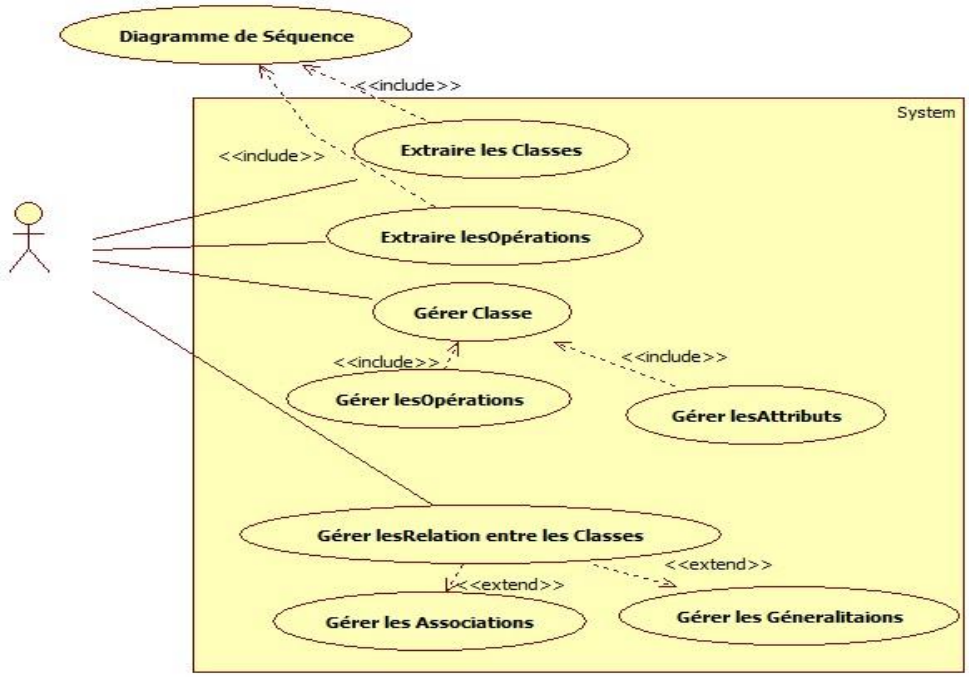


Fig.II.22. Diagramme des cas d'utilisation de diagramme de classe.

- **Description textuelle**

Titre : Gérer Classe.

Résumé : ce cas d'utilisation permet d'ajouter, modifier et supprimer les attributs de classe.

Acteur : utilisateur. **Responsable :**DjellouliSakina/Oukaci Fatiha

Pré condition : l'existence les diagrammes sur papier ou système modélisé.

Scénario nominal : Ajouter un Attribut de classe.

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3- Choisir le diagramme de classe.

4-Afficher la page de diagramme de classe.

5-Extraire une classe.

6-Choisir la page Opération.

7-Afficher les opérations de classe à sélectionner.

8-Choisir la page Attribut.

9-Entrer le nom d'Attribut.

10-Ajouter nom de classe.

11-Vérifier l'existence.

8-Afficher l'Attribut.

Scénario alternatif : Modifier et Supprimer une Attribut de classe.

Post condition : ensemble des classes correcte.

Titre : Gérer les relations entre les Classes.

Résumé : ce cas d'utilisation permet d'ajouter, modifier et supprimer les relations entre les Classes.

Acteur : utilisateur. **Responsable :**DjellouliSakina/Oukaci Fatiha

Pré condition : l'existence les diagrammes sur papier ou système modélisé.

Scénario nominal : Ajouter une Association.

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3- Choisir le diagramme de classe.

4-Afficher la page de diagramme de classe.

5-Choisir la page Extrémité l'Association.

6-Sélectionner une Classe1.

7-Sélectionner une Classe2.

8-Entrer le nom d'Association.

10-Ajouter nom d'Association.

11-Vérifier l'existence.

8-Afficher l'Association.

Scénario alternatif : Modifier et Supprimer une Association.

- **Modifier une Association :**

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3- Choisir le diagramme de classe.

4-Afficher la page de diagramme de classe.

5-Afficher les listes d'Association.

6-Sélectionner une Association.

7-Choisir la page Extrémité l'Association.

8-Modifier le nom de classe1.

9-Modifier le nom de classe2.

10-Modifier le nom d'Association

11-Vérifier l'existence.

12-Afficher l'Association.

- **Supprimer une Association :**

1- Accéder à l'application UML_MAC.

2-Afficher la page de diagramme de Cas d'Utilisation.

3- Choisir le diagramme de classe.

4-Afficher la page de diagramme de classe.

5-Afficher les listes d'Association.

6-Sélectionner une Association.

7-Choisir la page Extrémité l'Association.

8-Supprimer l'Association.

Post condition : ensemble des Associations correcte.

IV.2.2. Diagramme de cas d'utilisation de politique de sécurité

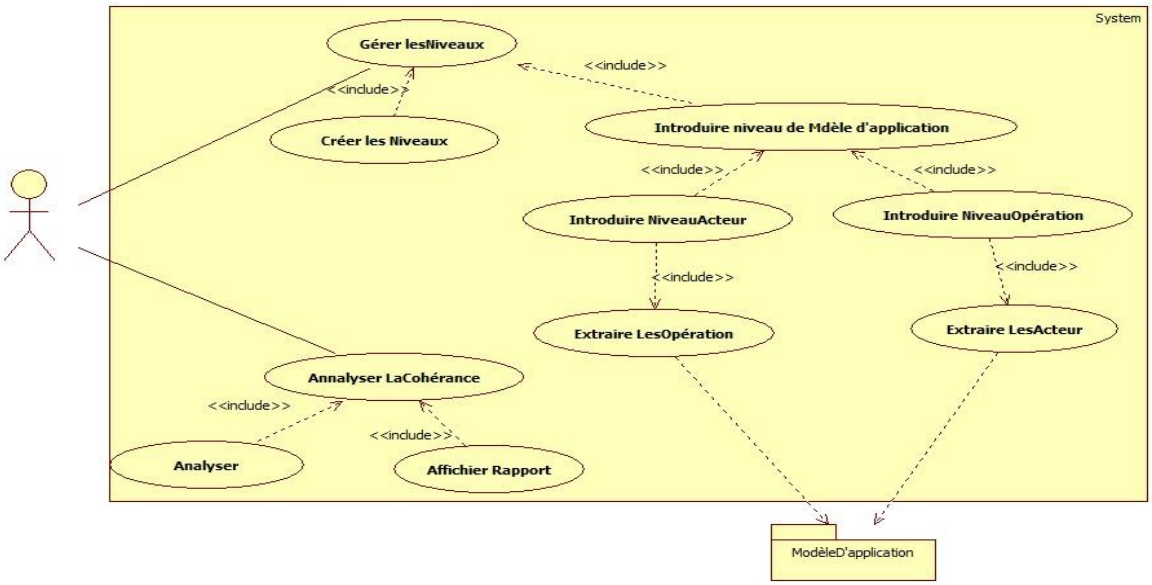


Fig.II.23. Vue de haut niveau des cas d'utilisation de Politique de sécurité.

IV.3. Les diagrammes de séquence d'analyse

- **Diagramme de séquence (Ajouter un Acteur)**

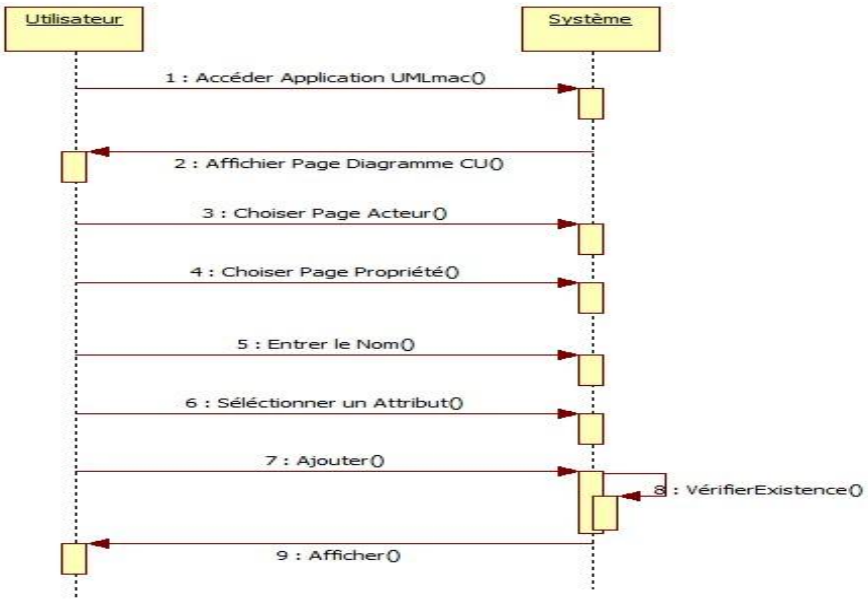


Fig.II.24. Diagramme de séquence (Ajouter un Acteur).

- **Diagramme de séquence modifier un Acteur**

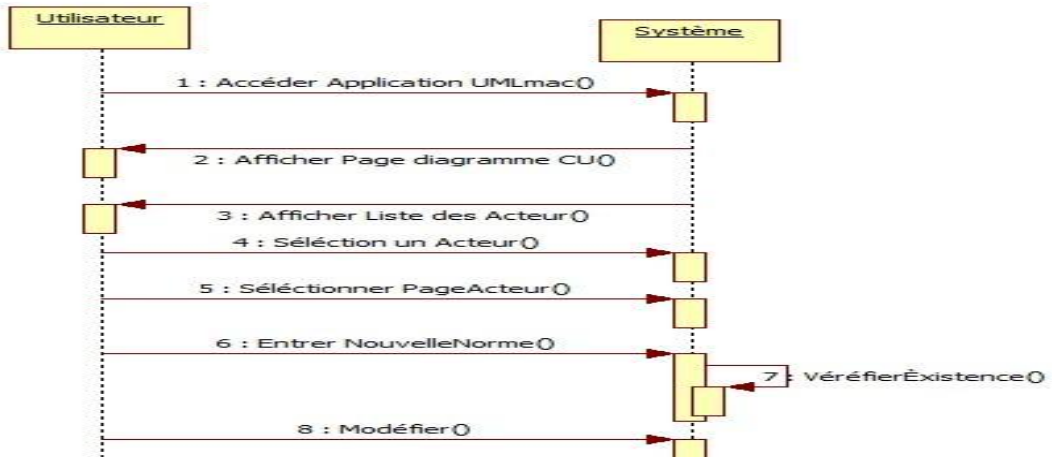


Fig.II.25. Diagramme de séquence (Modifier un Acteur).

- **Diagramme de séquence (Supprimer un Acteur)**

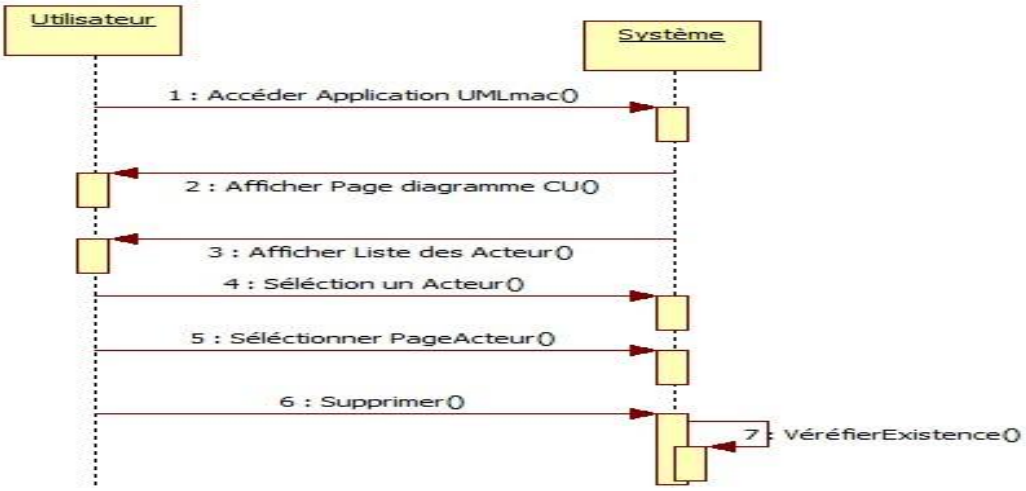


Fig.II.26. Diagramme de séquence (Supprimer un Acteur).

- **Diagramme de séquence (Ajouter un Cas d'Utilisation)**

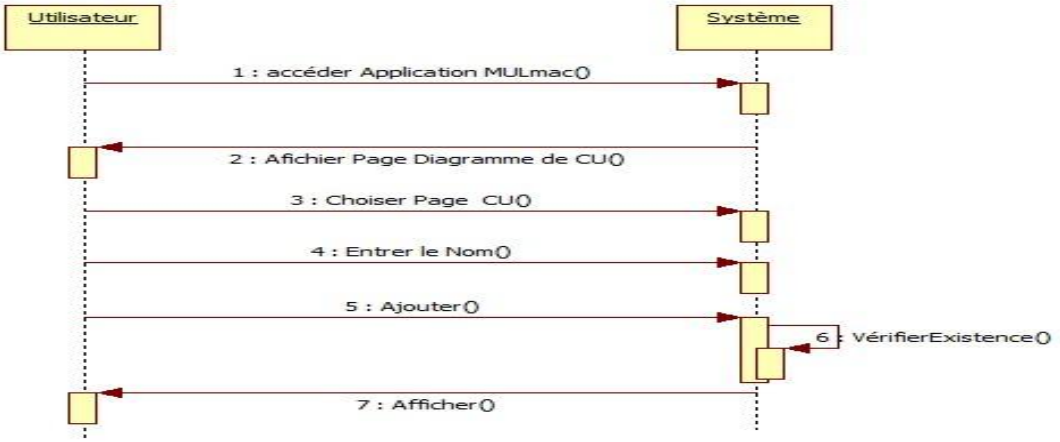


Fig.II.27. Diagramme de séquence (Ajouter un CU)

- **Diagramme de séquence (Supprimer un Cas d'Utilisation)**

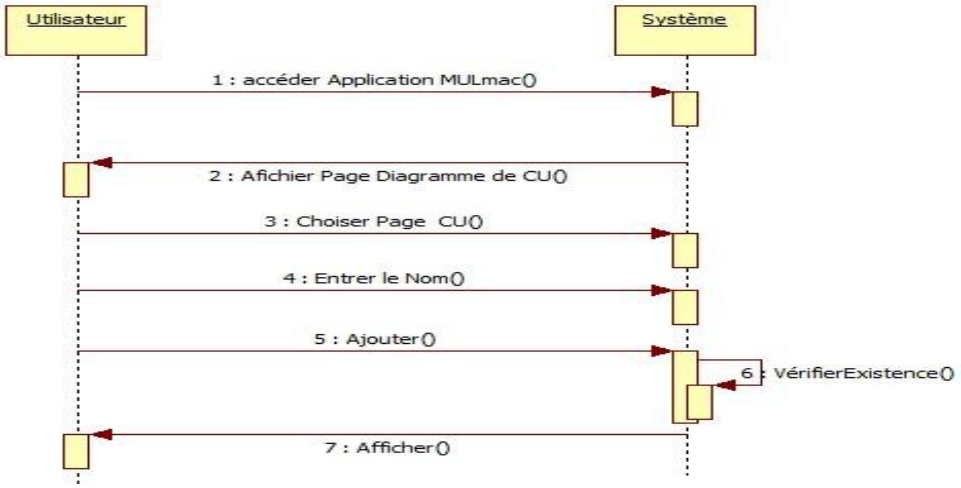


Fig.II.28. Diagramme de séquence (Supprimer un Cas d'Utilisation)

- **Diagramme de séquence modifier un cas d'utilisation**

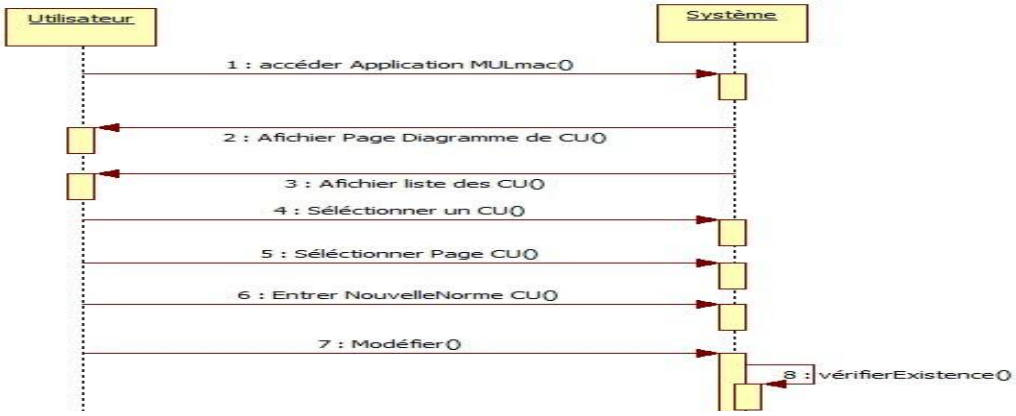


Fig.II.29. Diagramme de séquence (Modifier un cas d'utilisation).

- **Diagramme de séquence (Ajouter relation entre CU et Acteur)**

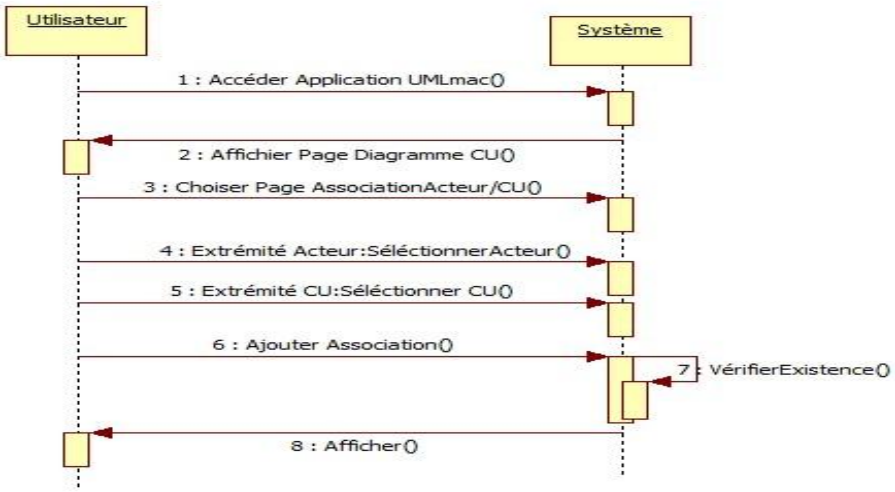


Fig.II.30. Diagramme de séquence (Ajouter relation entre CU et Acteur).

- **Diagramme de séquence (Ajouter relation entre Acteur)**

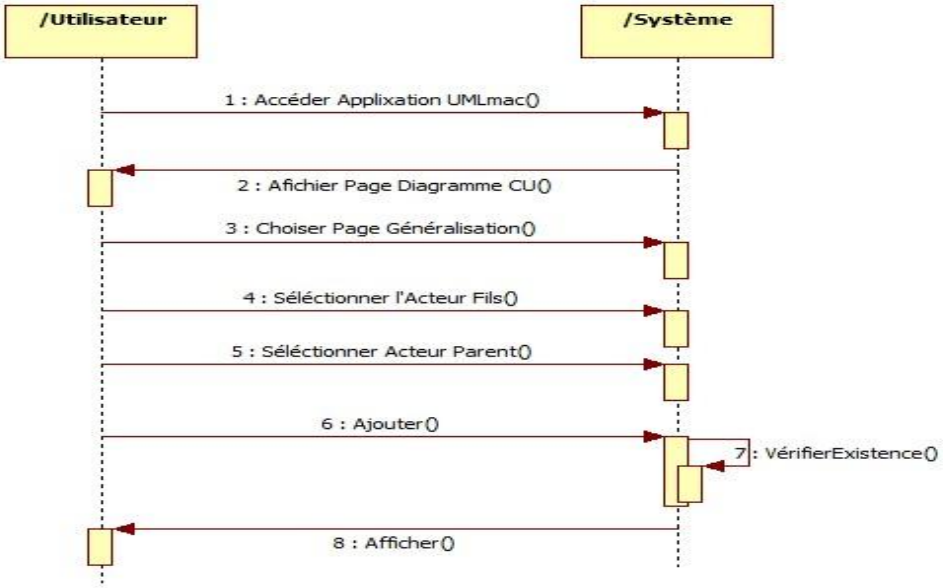


Fig.II. 31. Diagramme de séquence (Ajouter les relations entre Acteur).

- **Diagramme de séquence (Ajouter l'ObjetClasse)**

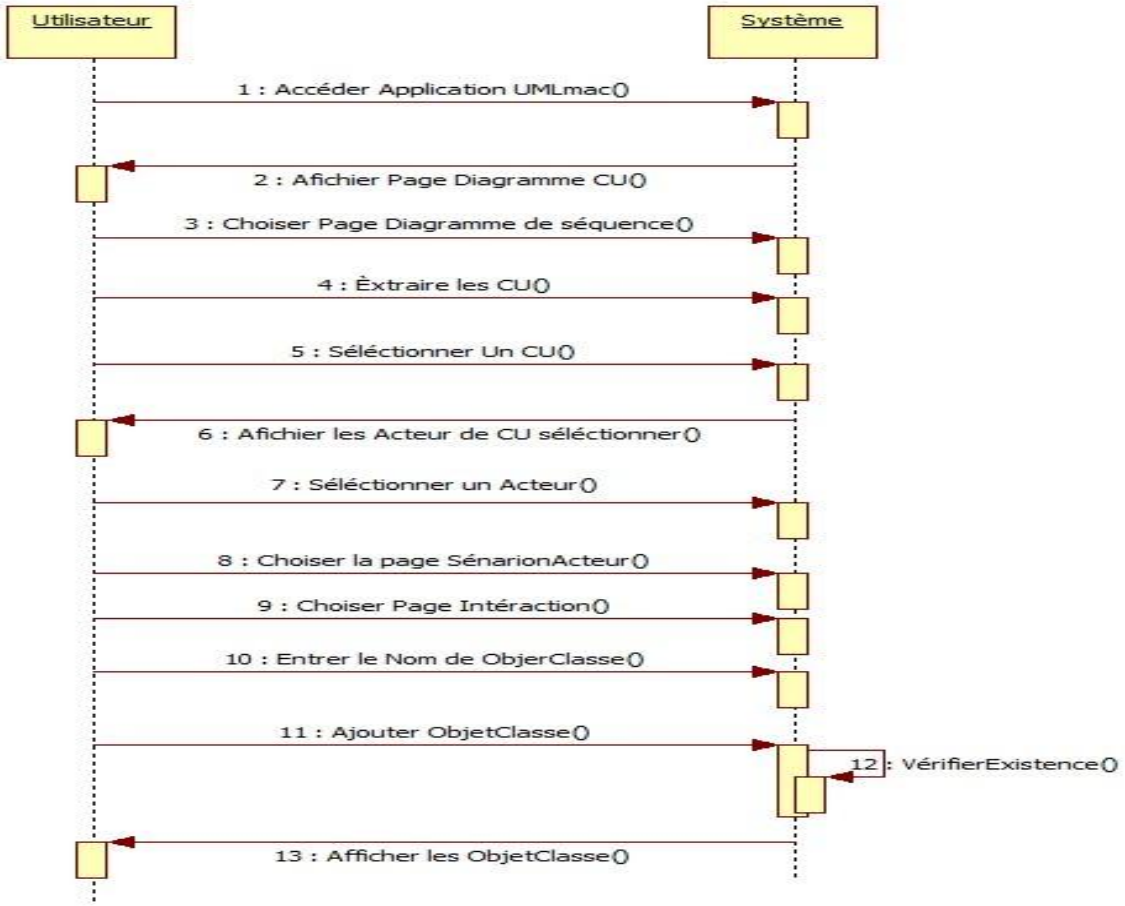


Fig.II.32. Diagramme de séquence (Ajouter l'ObjetClasse).

- **Diagramme de séquence (Introduire les Messages)**

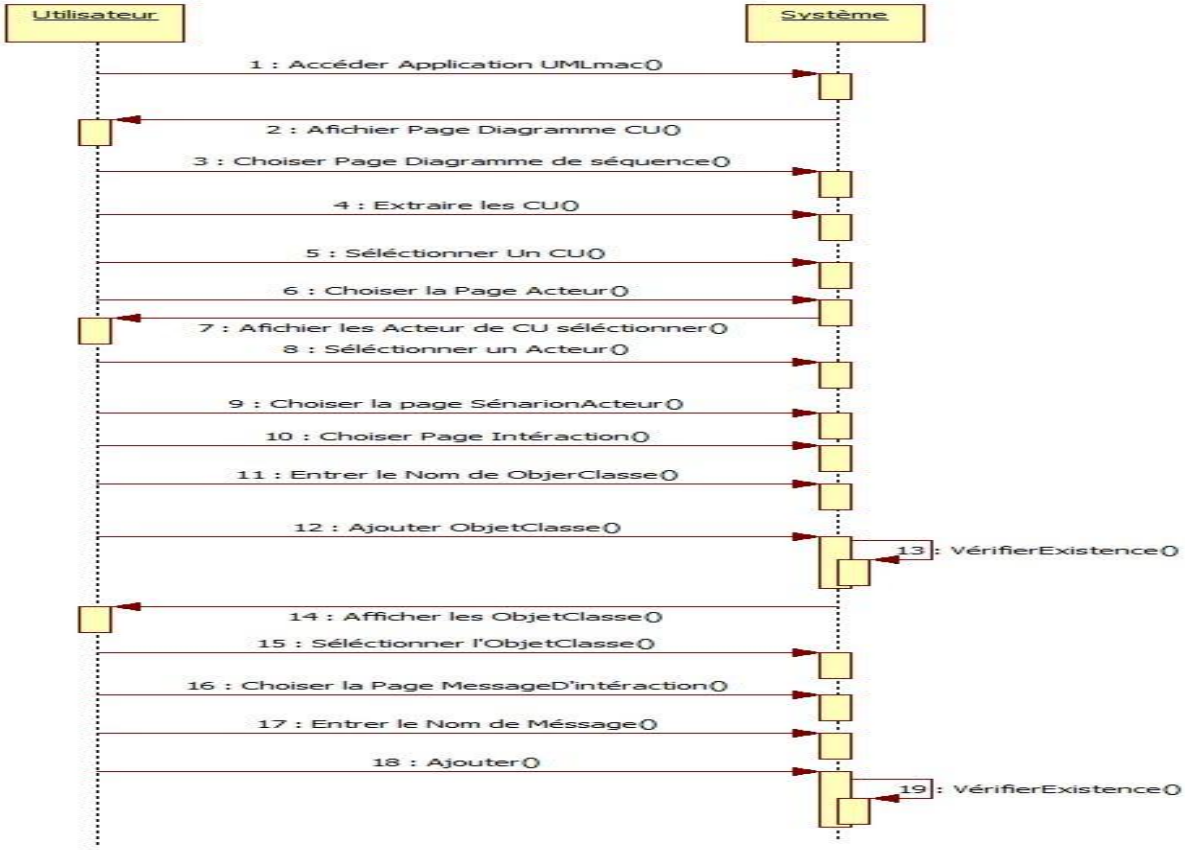


Fig.II.33. Diagramme de séquence D'introduire les scénarios de CU.

- **Diagramme de séquence (Gérer les Classes)**

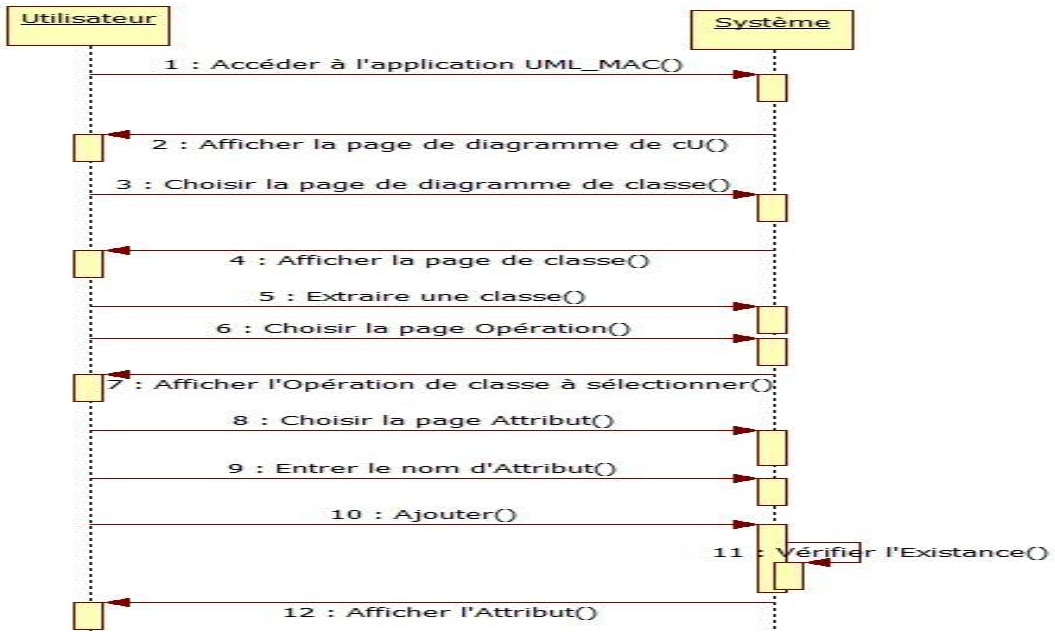


Fig.II.34. Diagramme de séquence (Gérer les Classes).

IV.4. La spécification des diagrammes de classe d'analyse

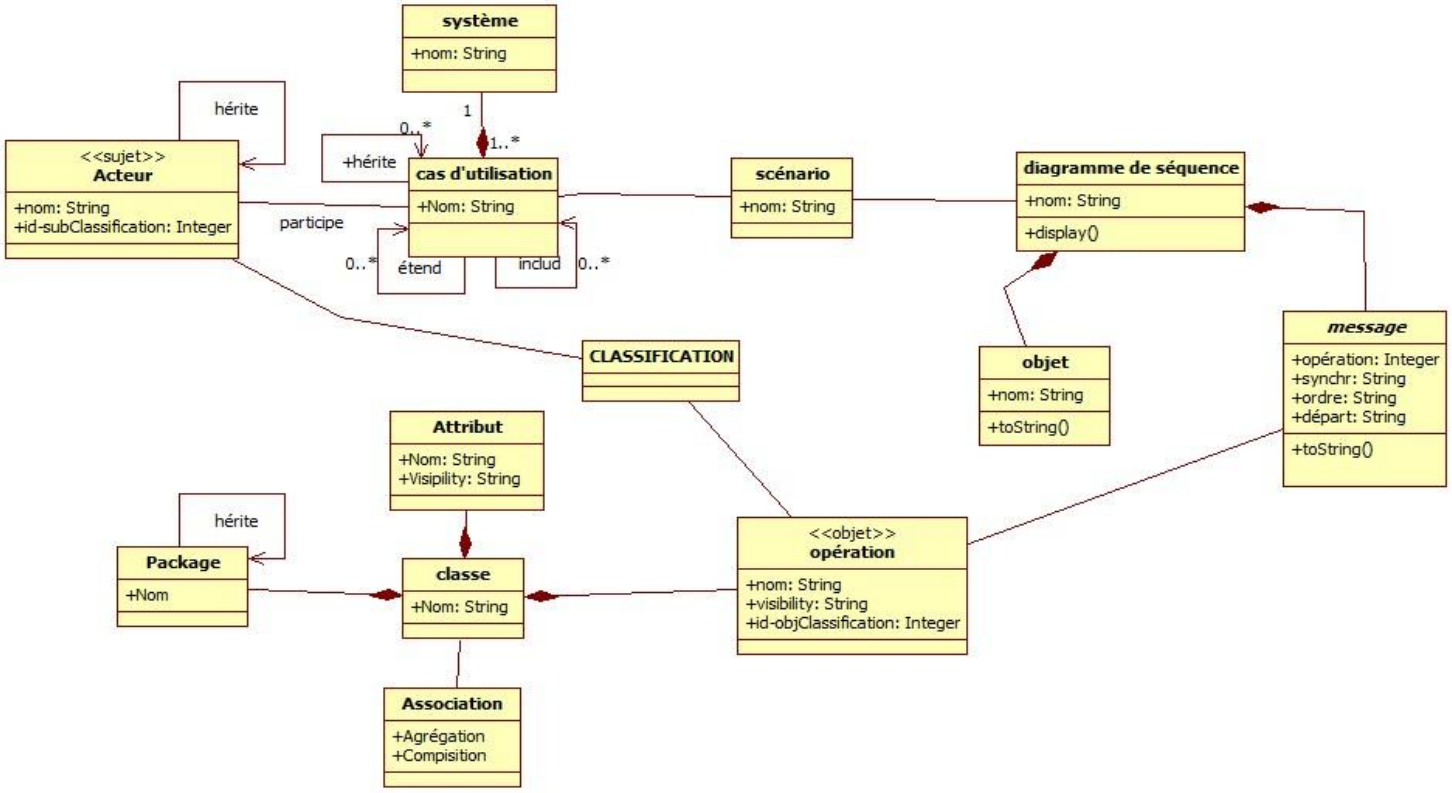


Fig.II.35. Diagramme de classe d'analyse.

V. Conception

L'analyse est le processus qui, en partant de l'examen des besoins, produit un modèle conceptuel du système à développer. Les classes d'analyse peuvent être réparties en trois catégories :

Les classes Interface : elles représentent les objets des interfaces entre utilisateur et système.

Les classes Contrôle : elles représentent les objets qui ont des activités système assez significatives pour être nommées, telles que les suppressions et les modifications…

Les classe Entité : ce sont les objets décrits dans les cas d'utilisation et qui représentent les objets persistants.

V.1. La spécification des diagrammesde Classes

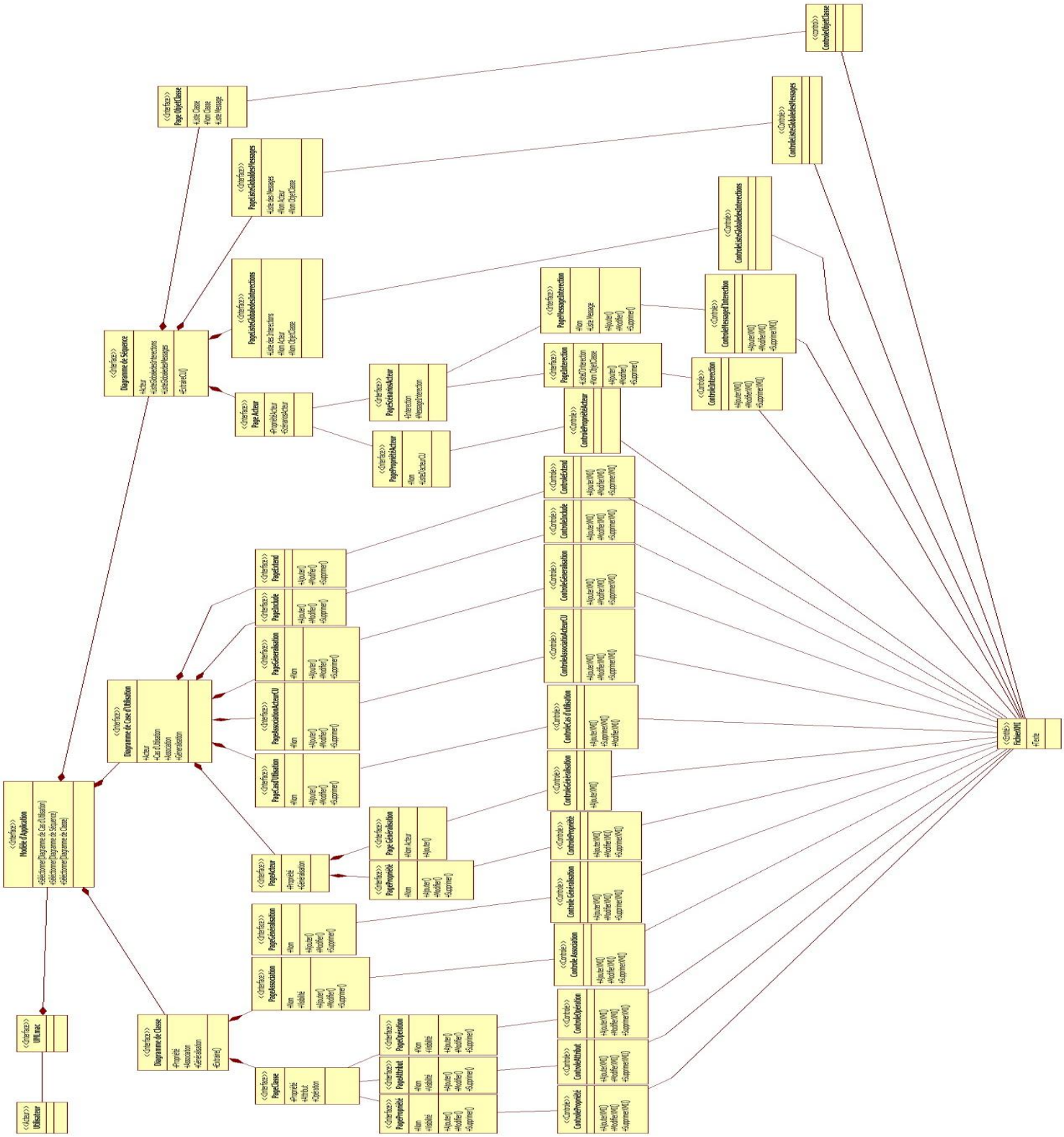


Fig.II.36. Diagramme de classe de conception de modèle d'application.

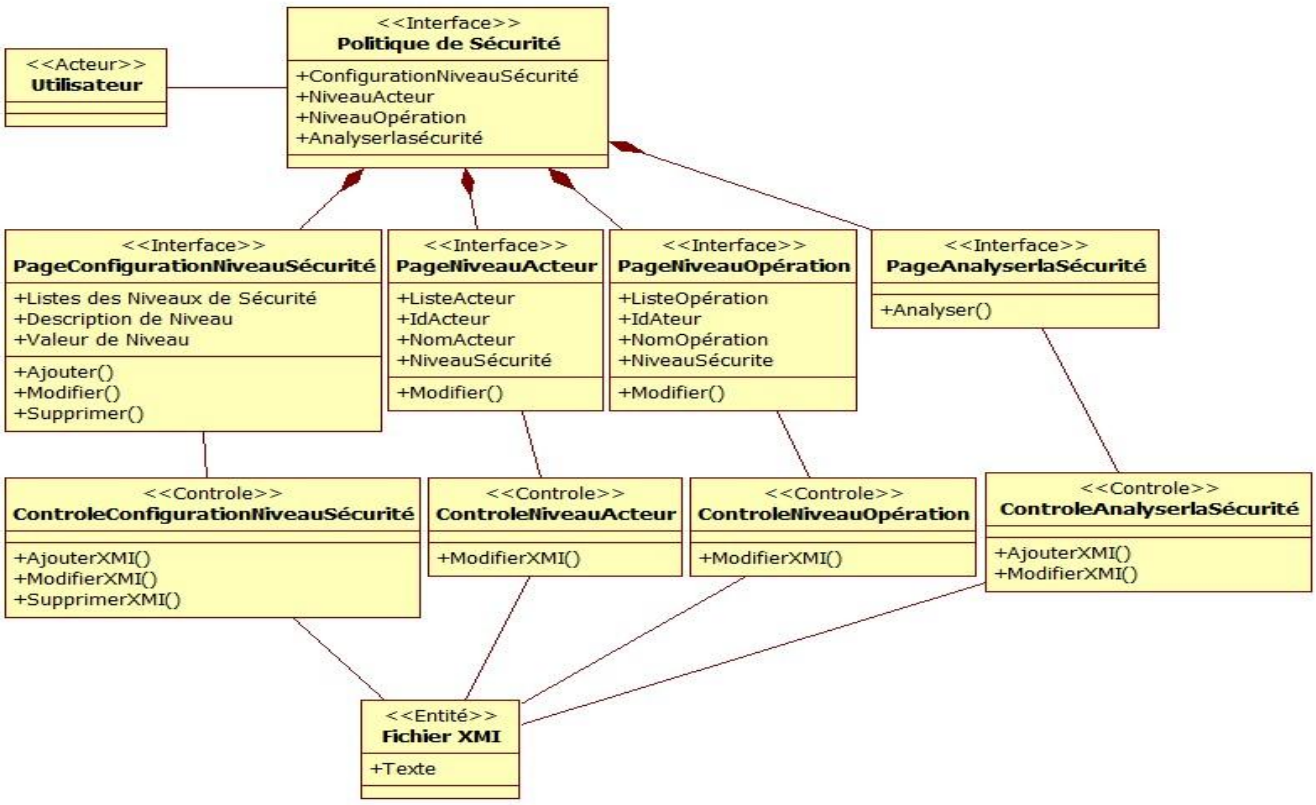


Fig.II.37. Diagramme de classe de conception de politique de sécurité.

V.2. La spécification des diagrammes de séquence

Dans cette partie de l'analyse nous avons généré les diagrammes de séquence à partir des diagrammes de classes d'analyse présentés ci-dessus.

Les diagrammes de séquence donnent une représentation dynamique décrivant toutes les interactions possibles entre l'utilisateur et les différents éléments de notre application qui sont décrits comme un ensemble d'objets d'analyse.

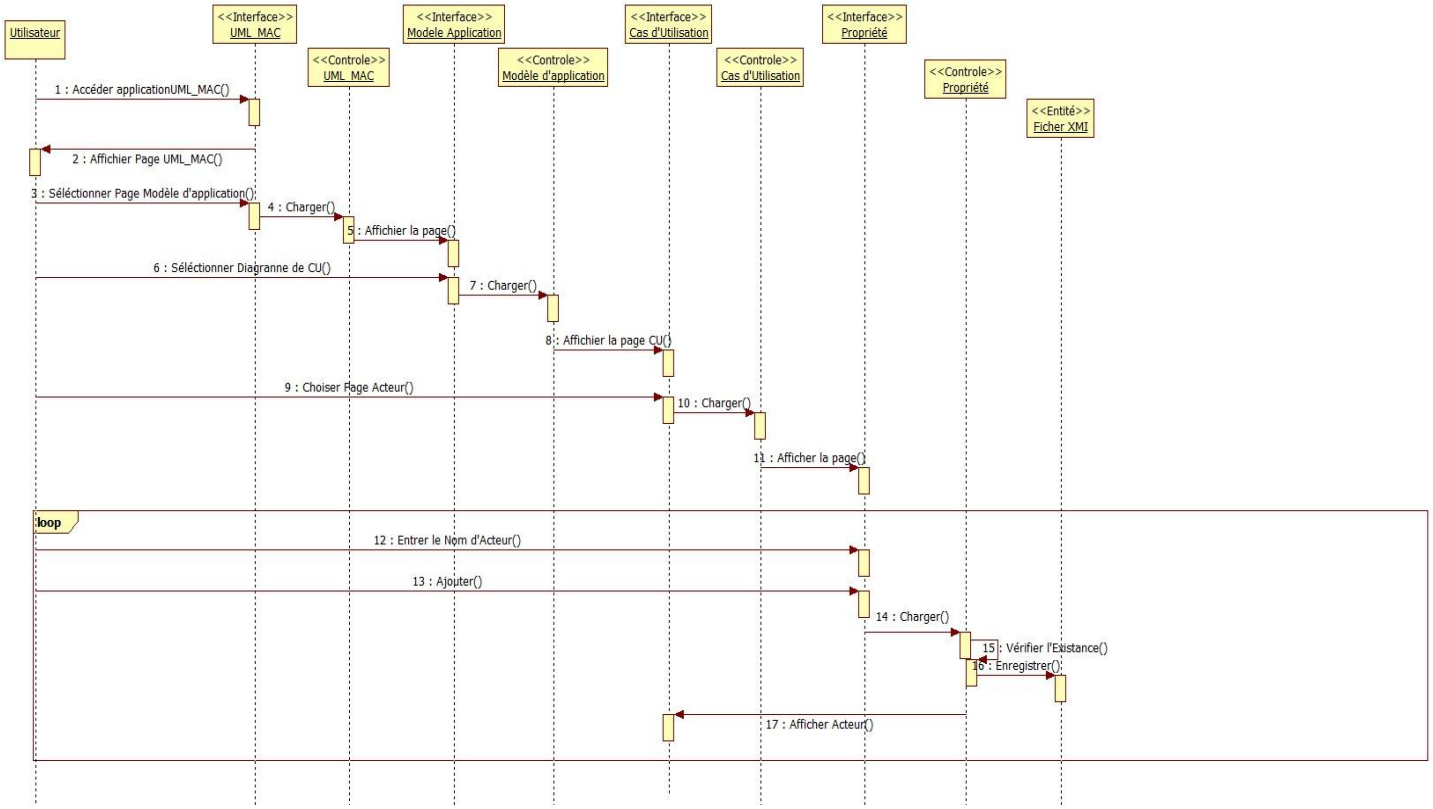


Fig.II.38. Diagramme de séquence de conception (Ajouter Acteur).

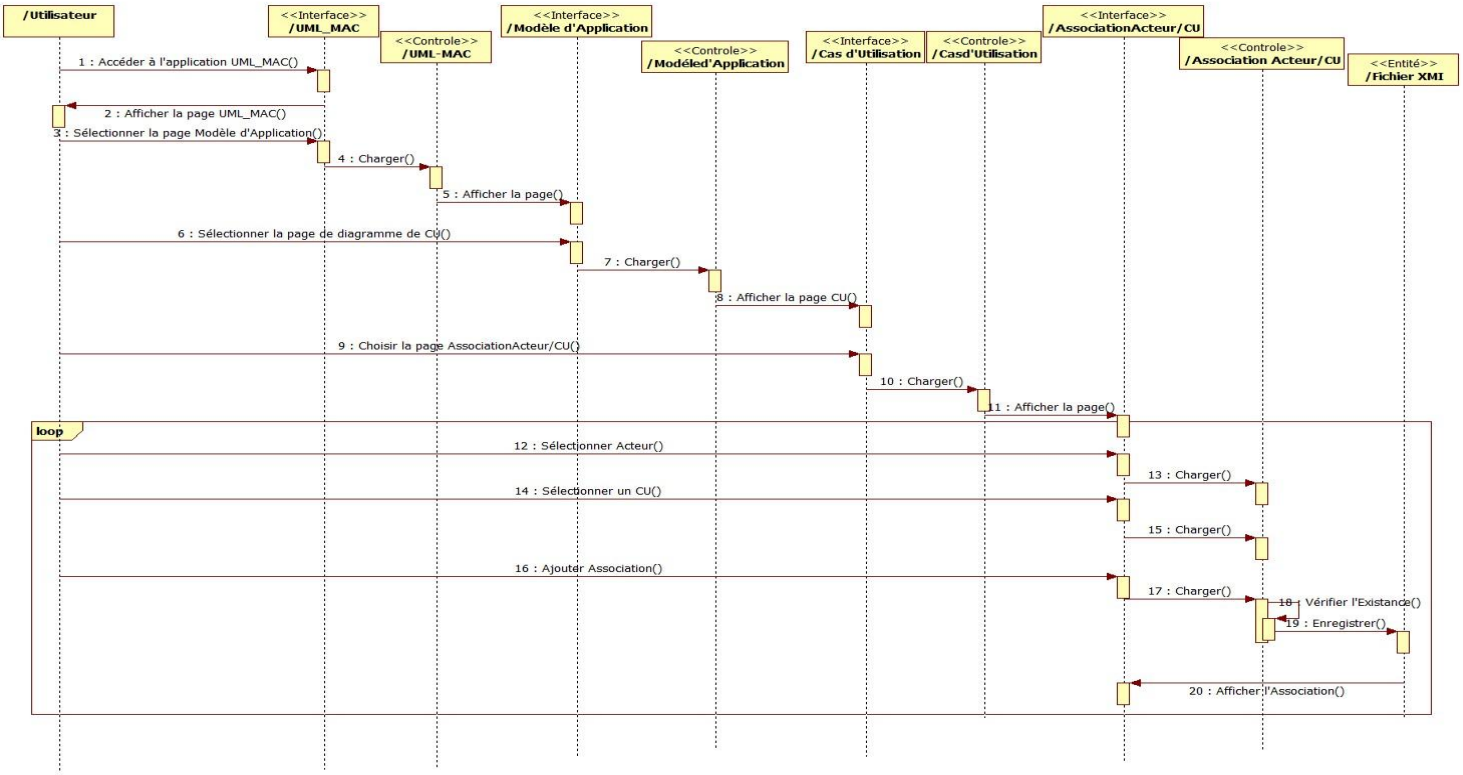


Fig.II.39. Diagramme de séquence de conception (Ajouter une relation entre Acteur et CU).

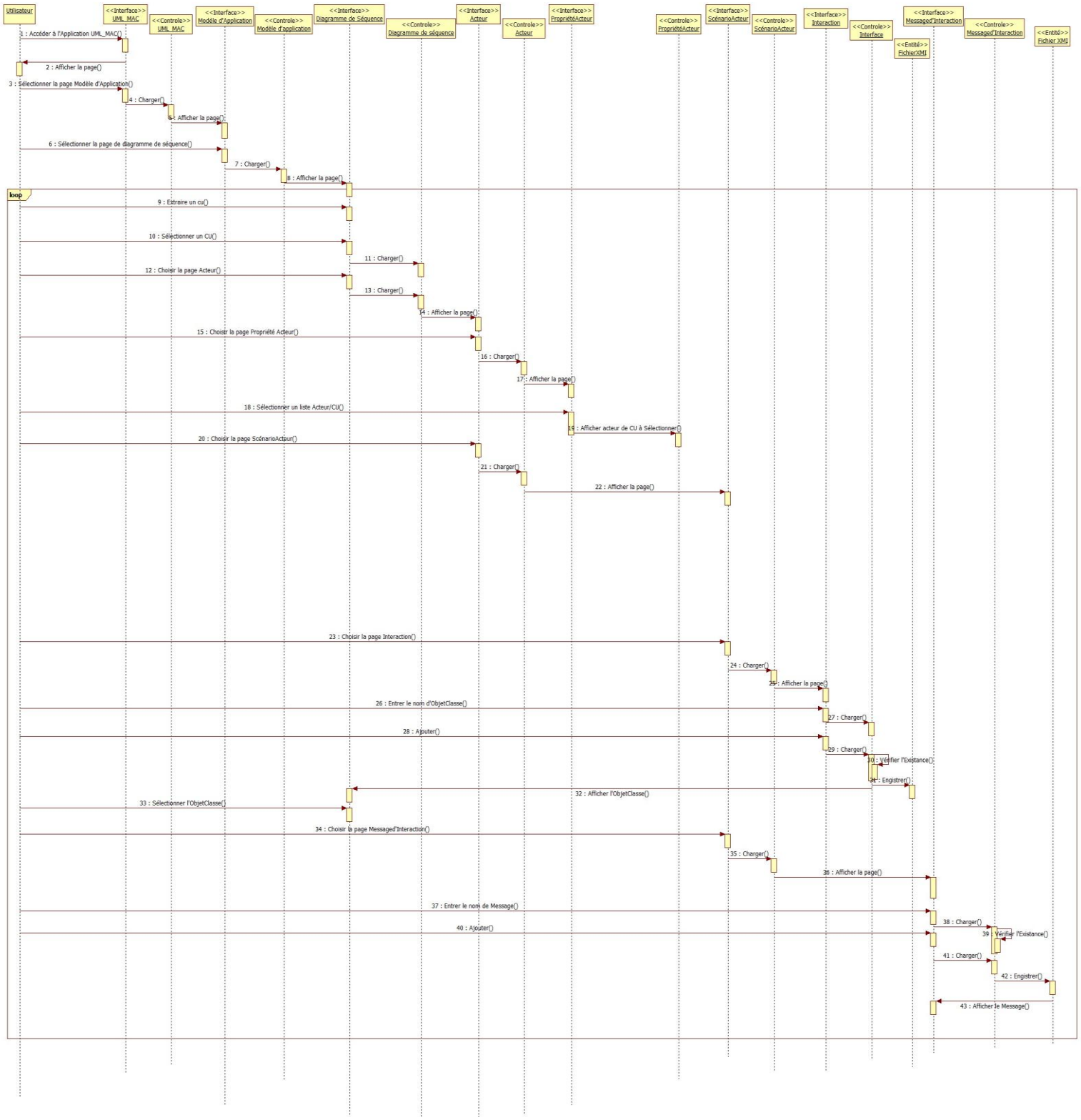


Fig.II.40. Diagramme de conception (Ajouter un message).

V.3. La spécification des diagrammes d'activités

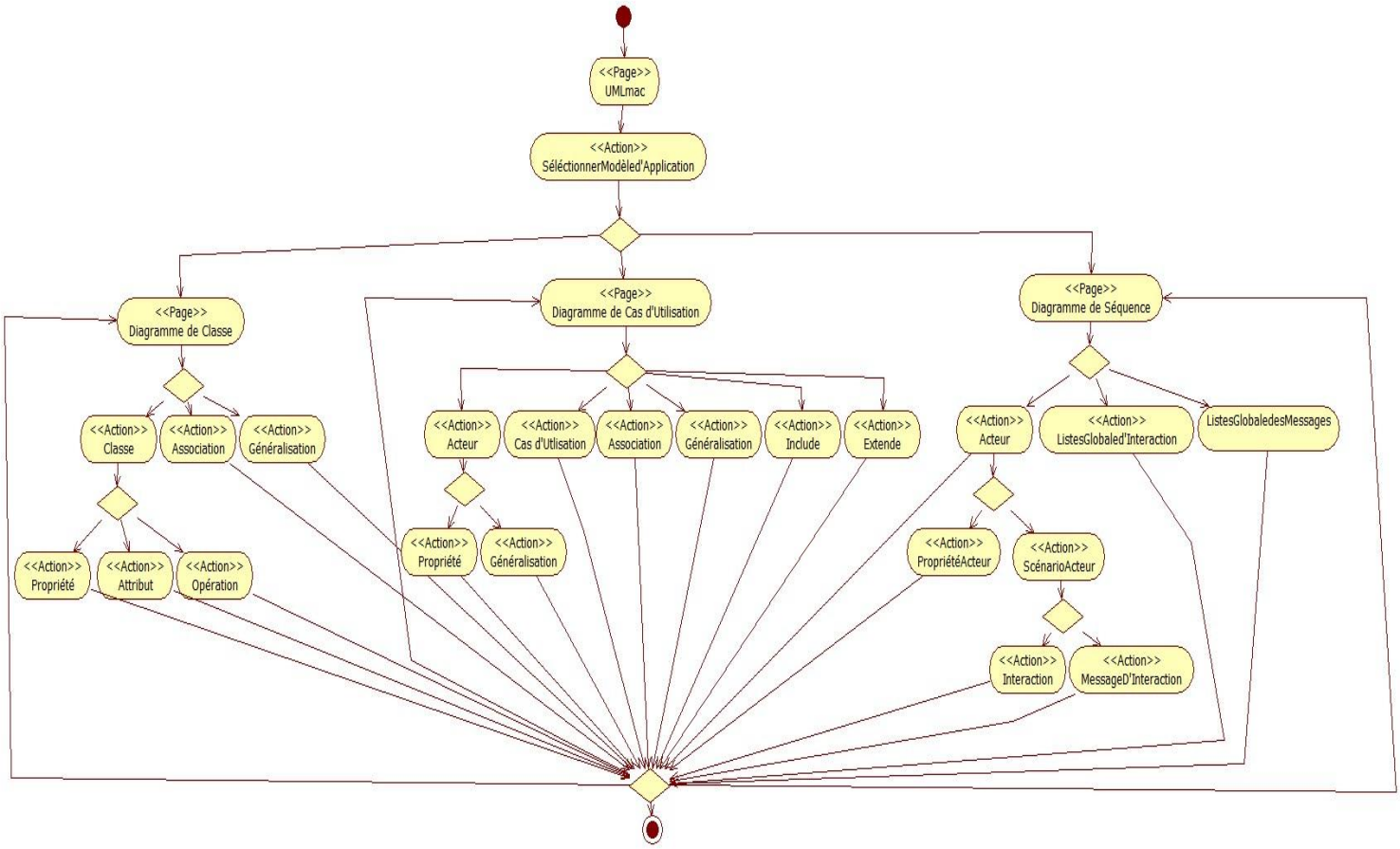


Fig.II.41. Diagramme d'activité de modèle d'application.

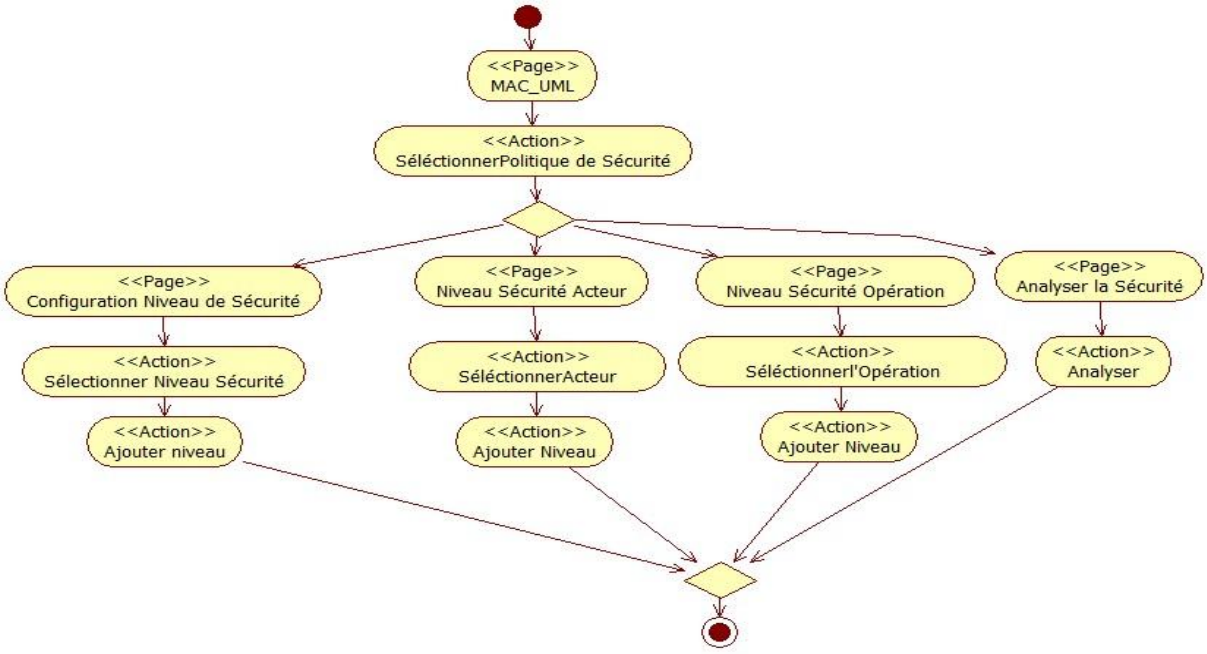


Fig.II.42. Diagramme d'activité de politique de sécurité.

V.4. Le diagramme de composants du système

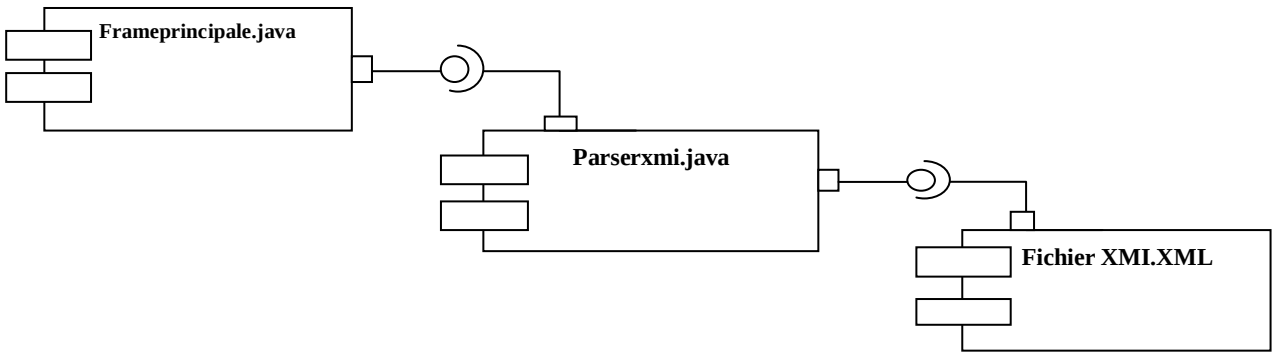


Fig.II.43. Diagramme de composant du système.

VI. Conclusion

Dans ce chapitre nous avons modélisé notre application avec le langage UML en utilisant le processus RUP, cette modélisation nous a permis de bien comprendre l'application ainsi ce la nous permet de bien préparer la phase d'implémentation.

Dans le chapitre suivant (implémentation) nous allons nous basé sur les diagrammes générés dans ce chapitre pour programmer notre application.

Chapitre III
L'implémentation

I. Introduction

Notre application est composé de deux partie, la première partie consiste à introduire le modèle d'application (les diagrammes UML) alors que la deuxième partie consiste à détecter les défiances de sécurité dans le modèle d'application, pour cela on a réparti notre application en deux modules, un module pour la définition et la manipulation des éléments de diagrammes UML (cas d'utilisation, acteur, association, classe, opération …) et un module pour la détection des défiances de sécurité de modèle d'application.

Ces modules enregistre et manipule les donnée (modèle d'application) dans une structure spéciale, cette structure peut être un fichier texte, cette approche est très difficile à introduire et à manipuler, une base de données, ce moyen de stockage demande l'installation d'un SGBD et en plus il est difficile à introduire et à manipuler ou un fichier XML, cette méthode est difficile a introduire mais elle est facile à manipuler.

Dans notre application on va utiliser un fichier XML pour enregistrer les éléments de modèle UML et précisément XMI (XML MetadataInterchange) qui est un standard pour l'échange d'informations de métadonnées UML basé sur XML.

II. Les outils utilisés pour réaliser l'application

De nos jours il existe de nombreux environnements de programmation et des pilotes pour les accès aux données, plus au moins dédiés à tel ou tel type d'applications particulières. Parmi eux, notre choix s'est focalisé vers le langage JAVA avec l'IDE NetBeans.

VI.1.1. Présentation de l'environnement JAVA

Java est à la fois un langage de programmation et un environnement d'exécution. Le langage Java a la particularité principale d'être portable sur plusieurs systèmes d'exploitation tels qu'UNIX, Microsoft Windows, Mac OS ou Linux... C'est la plateforme qui garantit la portabilité des applications développées en Java**[19]**.

VI.2. L'IDE NetBeans

NetBeans IDE est un environnement de développement intégré (EDI) modulaire basé sur des normes, Le projet de NetBeans IDE consiste en un EDI Open Source complet écrit dans le langage de programmation Java et en une plateforme d'application cliente riche, qui peut être utilisée comme structure générique pour créer n'importe quel type d'application**[20]**.

VI.3. Extensible MarkupLanguage XML

Le formalisme XML (Extended MarkupLanguage) permet la représentation structurée d'informations dans un format texte. Il peut par conséquent être utilisé comme format syntaxique de transport de modèles et de méta-modèles. La structure de l'information est alors définie dans un fichier annexe au format DTD (Document Type Description). L'avantage d'un tel formalisme est de permettre l'échange d'informations dès lors que l'on base celle-ci sur une DTD normalisée**[19]**.

Voici un exemple de document XML représentant un diagramme de classe :

```
<?xml version="1.0" ?>
<!DOCTYPE Livre "Livre.dtd">
<Livre Auteur="Michel Nakhlé et Charles Modiguy">
<Titre>Rapport</ Titre >
<Chapitre id="1">
        Premier Chapitre. Introduction.
</Chapitre >
<Chapitre id="2">
</Chapitre >
</Livre>
```

Dans ce formalisme, nous trouvons la notion d'élément. Les éléments sont définis avec un marqueur de début et un marqueur de fin. Le marqueur de début constitué simplement du nom de l'élément tandis que le marqueur de fin est constitué du nom de l'élément préfixé du caractère "/".

Dans l'exemple précédent, nous avons les éléments :

- Rapport représentant les rapports,
- Titre représentant le titre des rapports et
- Chapitre représentant l'intitulé d'un chapitre.

Ces éléments peuvent ensuite disposer d'attributs. Les attributs correspondent généralement au niveau le plus fin de décomposition des éléments. Dans l'exemple précédent, nous avons l'attribut Auteur défini sur l'élément Rapport et représentant le nom de l'auteur du rapport et l'attribut id sur l'élément Chapitre indiquant le numéro du chapitre

Le rôle du fichier au format DTD est de décrire tous les éléments que l'on est susceptible de rencontrer dans le fichier XML ainsi que tous les attributs que ces éléments sont susceptibles de disposer.

Actuellement, l'utilisation de DTD pour décrire le contenu des fichiers XML est en train de disparaître au profit des schémas. Les schéma sont des fichiers au format XML (ce qui n'était pas le cas des fichiers DTD) et proposent un certain nombre de facilités pour la définition de la structure des fichiers XML.

L'un des avantages de XML est sa souplesse et sa lisibilité tandis que ses désavantages sont l'aspect « verbeux » du langage entraînant rapidement des fichiers d'une taille importante et le fait que les données représentées dans un fichier XML le sont sous la forme d'un arbre. En effet dès que l'on souhaite représenter des graphes dans un document XML, il est nécessaire de définir des références qui vont alors à l'encontre de la souplesse et la lisibilité. Or les modèles et les méta-modèles sont plus généralement constitués de graphes que d'arbres**[20]**.

VI.4. XML MetadataInterchange : XMI

VI.4.1. Formalisme XMI

XMI est le langage d'échange entre le monde des modèles et le monde XML (eXtensibleMarkupLanguage). C'est le format d'échange standard entre les outils compatibles MDA. XMI décrit comment utiliser les balises XML pour représenter un modèle UML en XML. Cette représentation facilite les échanges de données (ou métadonnées) entre les différents outils ou plates-formes de modélisation. En effet, XMI définit des règles permettant de construire des DTD* (Document Type Définition) et des schémas XML à partir de méta-modèle, et inversement. Ainsi, il est possible d'encoder un méta-modèle dans un document XML, mais aussi, à partir de document XML il est possible de reconstruire des méta modèles. Les méta-modèles MOF et UML sont décrits par des DTD et les modèles traduits dans des documents XML conformes à leur DTD correspondante. XMI a l'avantage de regrouper les métadonnées et les instances dans un même document ce qui permet à une application de comprendre les instances grâce à leurs méta-données. Ceci facilite les échanges entre applications **[21]**.

VI.4.2. Objectifs XMI

L'objet principal de XMI est de permettre l'échange de méta données entre outils de modélisation basés sur UML et la communication des répertoires de méta données basés sur le MOF deux standards de l'OMG.

L'effet économique et technique est important, parce que jusqu'ici il était à peu près impossible à une communauté de travail, un groupe d'entreprises… de travailler sur des modèles communs en utilisant des outils de conception provenant de fournisseurs différents.

XMI se fonde sur les trois standards XML, UML et MOF :

- UML est un formalisme orienté objet de conception et de documentation d'applications,
- MOF est une technologie de définition et de représentation de méta données en tant qu'objets CORBA. Pour mémoire, CORBA est un concept permettant à deux applications de communiquer entre elles sans se soucier de leur localisation ou de leur mode de conception.

Les spécifications de XMI consistent à proposer un ensemble de règles de transformation de structures MOF en DTD XML, des DTD pour UML et MOF, un ensemble de règles de production de documents XML pour manipuler des méta données MOF. XMI est donc la rencontre entre deux mondes importants, celui des objets et celui des structures de données et documents **[21]**.

VI.4.3. Exemple fichier XMI généré à partir de notre application

En guise d'exemple, pour bien comprendre le formalisme, voyons ce fichier XMI, qui représente un diagramme de cas d'utilisation avec un acteur «factorier» et un cas d'utilisation «Ajouter produit».

```
- <UML:Actor xmi.id="2" name="factorie" isSpecification="false" isRoot="false" isLeaf="false" isAbstract="false" isActive="false" niveau
  - <UML:GeneralizableElement.generalization>
      <UML:Generalization xmi.idref="9"/>
    </UML:GeneralizableElement.generalization>
  </UML:Actor>
- <UML:Actor xmi.id="3" name="chef_factorie" isSpecification="false" isRoot="false" isLeaf="false" isAbstract="false" isActive="false" n
    <UML:GeneralizableElement.generalization/>
  </UML:Actor>
  <UML:UseCase xmi.id="4" name="Ajouter produit" isSpecification="false" isRoot="false" isLeaf="false" isAbstract="false" isActive="f
  <UML:UseCase xmi.id="5" name="Chercher produit" isSpecification="false" isRoot="false" isLeaf="false" isAbstract="false" isActive=
  <UML:UseCase xmi.id="6" name="Virifier produit" isSpecification="false" isRoot="false" isLeaf="false" isAbstract="false" isActive="fa
```

VI.4.4. Le parsing d'un fichier XML

Ile est essentiel pour le receveur d'un document XML de pouvoir extraire les données du document. Cette opération est possible à l'aide d'un outil appelé **parseur.**

Le parseur permet de créer une structure hiérarchique contenant les données contenues dans le document XML.

VI.5. L'API JDOM

JDOM est une API open source Java dont le but est de représenter et manipuler un document XML de manière intuitive pour un développeur Java sans requérir une connaissance pointue de XML. Par exemple, JDOM utilise des classes plutôt que des interfaces. Ainsi pour créer un nouvel élément, il faut simplement instancier une classe.

Malgré la similitude de nom entre JDOM et DOM, ces deux API sont très différentes. JDOM est une API uniquement Java car elle s'appuie sur un ensemble de classes de l'API Java notamment celles de l'API Collection **[22]**.

VI.5.1. La présentation de JDOM

Le but de JDOM n'est pas de définir un nouveau type de parseur mais de faciliter la manipulation au sens large de document XML : lecture d'un document, représentation sous forme d'arborescence, manipulation de cet arbre, définition d'un nouveau document, exportation vers plusieurs formats cibles ...

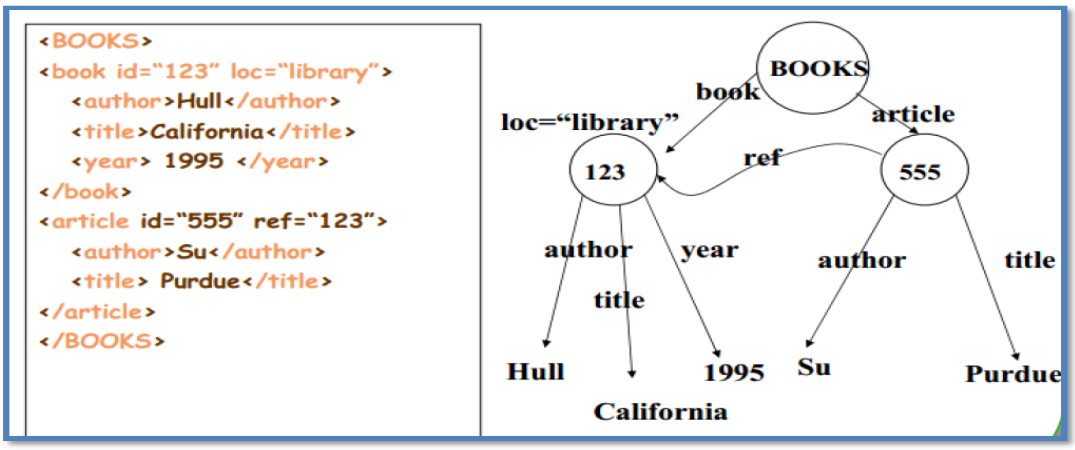


Fig.III.44. JDOM Représentation arborescente d'un document XML.

Dans le rôle de manipulation sous forme d'arbre, JDOM possède moins de fonctionnalités que DOM mais en contrepartie il offre une plus grande facilité pour répondre aux cas les plus classiques d'utilisation.
Cette facilité d'utilisation de JDOM lui permet d'être une API dont l'utilisation est assez répandue.

JDOM est donc un modèle de documents objets open source dédié à Java pour encapsuler un document XML. JDOM propose aussi une intégration de SAX, DOM, XSLT et XPath.

JDOM n'est pas un parseur : il a d'ailleurs besoin d'un parseur externe de type SAX ou DOM pour analyser un document et créer la hiérarchie d'objets relative à un document XML. L'utilisation d'un parseur de type SAX est recommandée car elle consomme moins de ressources que DOM pour cette opération. Par défaut, JDOM utilise le parseur défini via JAXP.

Un document XML est encapsulé dans un objet de type Document qui peut contenir des objets de type Comment, Processing Instruction et l'élément racine du document encapsulé dans un objet de type Element.

Fig.III.45. Méthodes de la classe Document.

Les éléments d'un document sont encapsulés dans des classes dédiées : Element, Attribute, Text, ProcessingInstruction, Namespace, Comment, DocType, EntityRef, CDATA.

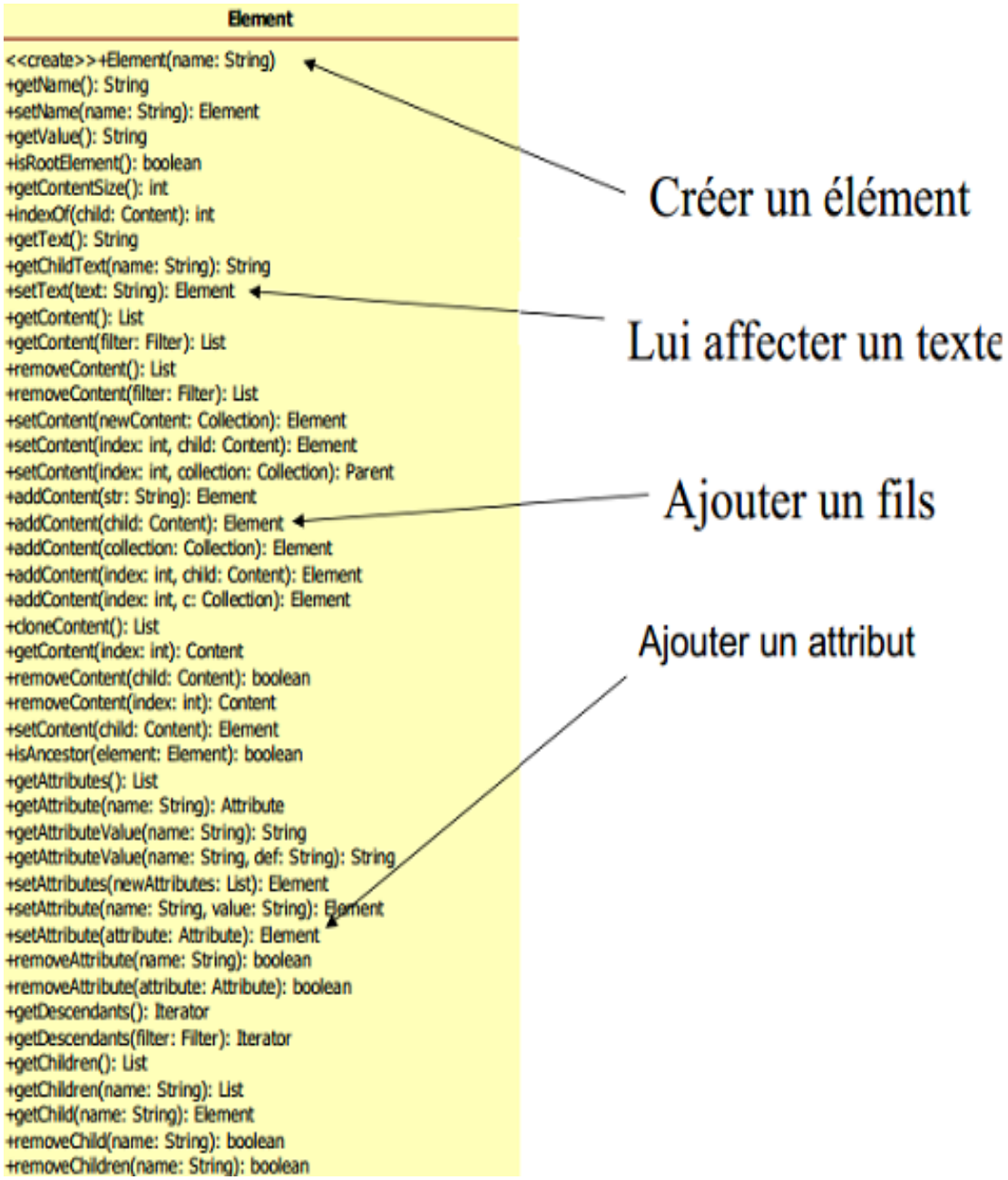


Fig.III.46. Méthodes de la classe Element.

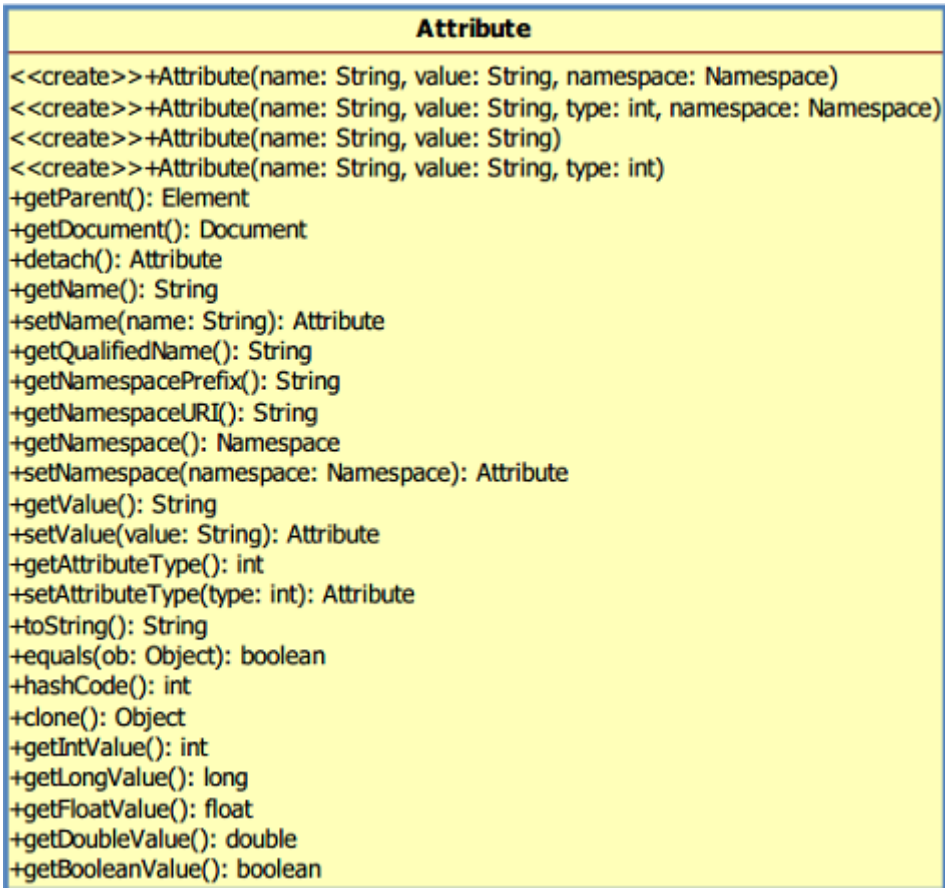


Fig.III.47. Méthodes de la classe Attribute.

Un objet de type Element peut contenir des objets de type Comment, Text et d'autres objets de type Element.

A l'exception des objets de type Namespace, les éléments sont créés en utilisant leur constructeur.

JDOM vérifie que les données contenues dans les éléments respectent la norme XML : par exemple, il n'est pas possible de créer un commentaire contenant deux caractères moins qui se suivent.

Une fois un document XML encapsulé dans un arbre d'objets, il est possible de modifier cet arbre dans le respect des spécifications de XML.

JDOM permet d'exporter un arbre d'objets d'un document XML dans un flux, un arbre DOM ou un ensemble d'événements SAX.

JDOM interagit donc avec SAX et DOM pour créer un document en utilisant ces parseurs ou pour exporter un document vers ces API, ce qui permet de facilement intégrer JDOM dans des traitements existants. JDOM propose cependant sa propre API.

VI.5.2. Les fonctionnalités et les caractéristiques

JDOM propose plusieurs fonctionnalités :

- Création de documents XML
- Encapsulation d'un document XML sous la forme d'objets Java de l'API
- Exportation d'un document dans un fichier, un flux SAX ou un arbre DOM
- Support de XSLT
- Support de XPath

Les points caractéristiques de l'API JDOM sont :

- elle est développée spécifiquement en et pour Java en utilisant les fonctionnalités de Java au niveau syntaxique et sémantique (utilisation des collections de Java 2, de l'opérateur new pour instancier des éléments, redéfinition des méthodes equals(), hashCode(), toString(), implémentation des interfaces Cloneable et Serializable, ...)

- elle se veut rapide et légère

  - elle veut masquer la complexité de certains aspects de XML tout en respectant ses spécifications
  - elle doit permettre les interactions entre SAX et DOM. JDOM peut encapsuler un document XML dans un hiérarchie d'objets à partir d'un flux, d'un arbre DOM ou d'événements SAX. Il est aussi capable d'exporter un document dans ces différents formats.

Il est légitime de se demander qu'elle est l'utilité de proposer une nouvelle API pour manipuler des documents XML en Java alors que plusieurs standards existent déjà. En fait le besoin est réel car JDOM propose des réponses à certaines faiblesses de SAX et DOM.

SAX est particulièrement bien adapté à la lecture rapide avec peu de ressources d'un document XML mais son modèle de traitement par événements n'est pas intuitive et surtout SAX ne permet pas de modifier ni de naviguer dans un document.

JDOM propose d'apporter une solution à ces différents problèmes dans une seule et même API **[22]**.

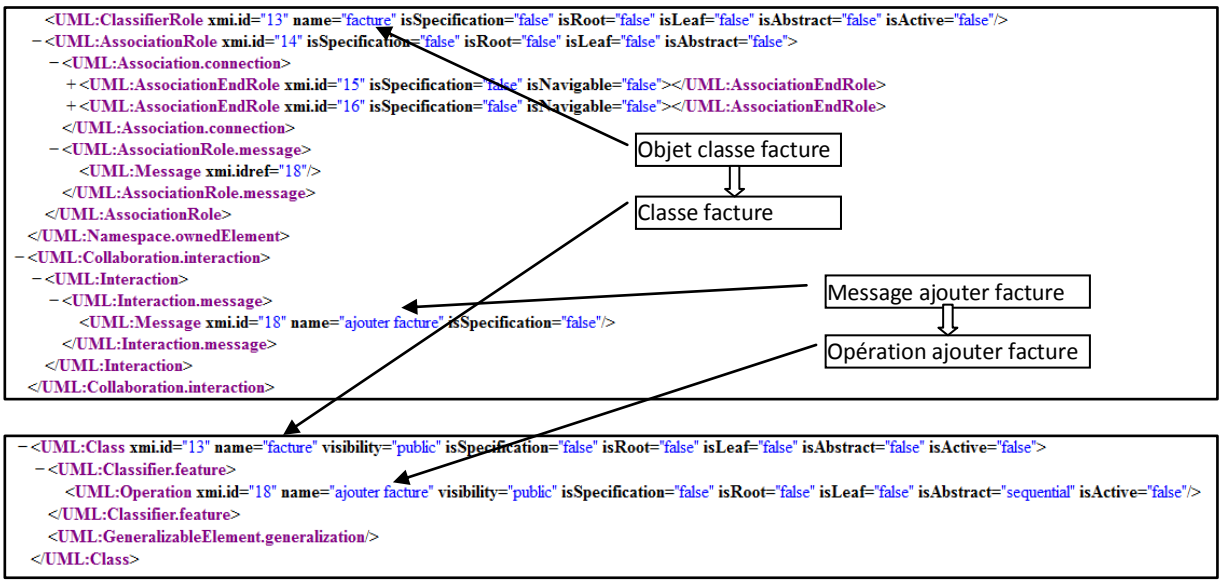
III. Présentation de quelque fonctions de notre application

- **Fonction enregistre(String fichier) :** cette fonction permet d'enregistrer les modifications sur un fichier xmi.

```
static void enregistre(String fichier)
{
try  {

      //********On utilise ici un affichage classique avec getPrettyFormat()*************
XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
      //******************** créer une instance de FileOutputStream***************
sortie.output(document, new FileOutputStream(fichier));
      }catch (java.io.IOException e)
{}
   }
```

- **Fonction extraireclass( ) :** cette fonction permet de parcourir tout les instances de classe et les messages de diagramme de séquence pour extraire et introduire les classes et opérations de diagramme de classe.



- **Code de fonction :**

```
privatevoidextraireclass()
{
int index =0;
   //****vérification si la liste des classes objet de diag de séquence est non vide *****
if(!listeelementfilsassrol2.isEmpty()) {

while(index < listeelementfilsassrol2.getSize() )
          {
if (Parserxmi.chekelem(Parserxmi.Name_space,
listeelementfilsassrol2.getElementAt(index).toString(), "Actor",1))
          {
 }
else
          {
      //**************Extraire les classes **********************************

            Element ele = (Element) listeelementfilsassrol2elm.getElementAt(index);
            String visibility  = null;

if (this.jRadioButtonclapac.isSelected()) { visibility = "packtage"; }
else if (this.jRadioButtonclapro.isSelected()) {   visibility = "protégé"; }
else if (this.jRadioButtonclapri.isSelected()) {    visibility = "privé";  }
else if (this.jRadioButtonclapub.isSelected()) {   visibility = "public";}
else visibility = "public";

      String isSpecification  = null;
isSpecification =  "false";
      String isRoot  = null;
if(jRadioButtonclaisr.isSelected()) isRoot= "true";
elseisRoot="false";
      String isLeaf  = null;
if(jRadioButtonclaisl.isSelected()) isLeaf= "true";
elseisLeaf="false";
      String isAbstract  = null;
if(jRadioButtonclaisa.isSelected()) isAbstract= "true";
elseisAbstract="false";
      String isActive  = null;
if(jRadioButtonclaisac.isSelected()) isActive= "true";
elseisActive="false";

if (Parserxmi.chekelem(Parserxmi.Name_space, ele.getAttributeValue("name"), "Class",0))
{
```

```
  }
else  {
     //**************************ajouter les classes ***********************
Parserxmi.ajoutElement1(Parserxmi.Name_space, "Class ,ele.getAttributeValue("xmi.id"));
    //*************************************enregistré le document xmi****
Parserxmi.enregistre("fichierxmi3.xmi");
      // ************************parcourire et ajouter les opération*************
DefaultListModel limes = (DefaultListModel)
Parserxmi.listeassroleclas(ele.getAttributeValue("xmi.id"));
int ii = 0;
while (ii<limes.getSize())
      {
Element el = (Element) limes.getElementAt(ii);
Parserxmi.listemessassorol(el);
ii++;
      }

DefaultListModellisteglobalmessage = Parserxmi.afficherFilsElement(Parserxmi.Mess_inter,
0,null,null, "Message");
DefaultListModelnamelistmessage = (DefaultListModel)
listeglobalmessage.getElementAt(0);
DefaultListModelidlistmessage = (DefaultListModel) listeglobalmessage.getElementAt(1);
DefaultListModelmymessage = new DefaultListModel();

intjij = 0;
while (jij<Parserxmi.Listmessobjetclass.size())
       {
int iii = 0;
while (iii<idlistmessage.size())
       {

if
(!idlistmessage.getElementAt(iii).toString().equals(Parserxmi.Listmessobjetclass.getElement
At(jij).toString()))
{
}
else
        {
          Element eleclas = (Element)
Parserxmi.getelem(ele.getAttributeValue("name"),"Class", Parserxmi.Name_space,0);
DefaultListModellisteelementope = Parserxmi.afficherFilsElement(eleclas, 0,null,null,
"Classifier.feature");
DefaultListModellisteelementfilsope = (DefaultListModel) listeelementope.getElementAt(6);
          Element var66 =  (Element) listeelementfilsope.getElementAt(0);
```

```
if (Parserxmi.chekelem(var66, namelistmessage.getElementAt(iii).toString(), "Operation",1))
{
 }
else   {
          // **********************ajouter les fonction a partir des message********
Parserxmi.ajoutElement1(var66, "Operation"
,melistmessage.getElementAt(iii).toString(),"public","false","false",
"false","sequential","false","0");
        // ************************enregistré le document xmi******************
Parserxmi.enregistre("fichierxmi3.xmi");
           }
  }
            iii++;
        }
jij++;
        }
}

}
           index++;
}
}
}
```

IV. Présentation des interfaces de notre application

Dans que ca suive nous allons présenter les déférentes interfaces de notre application

- **Interface de diagramme de cas d'utilisation :**

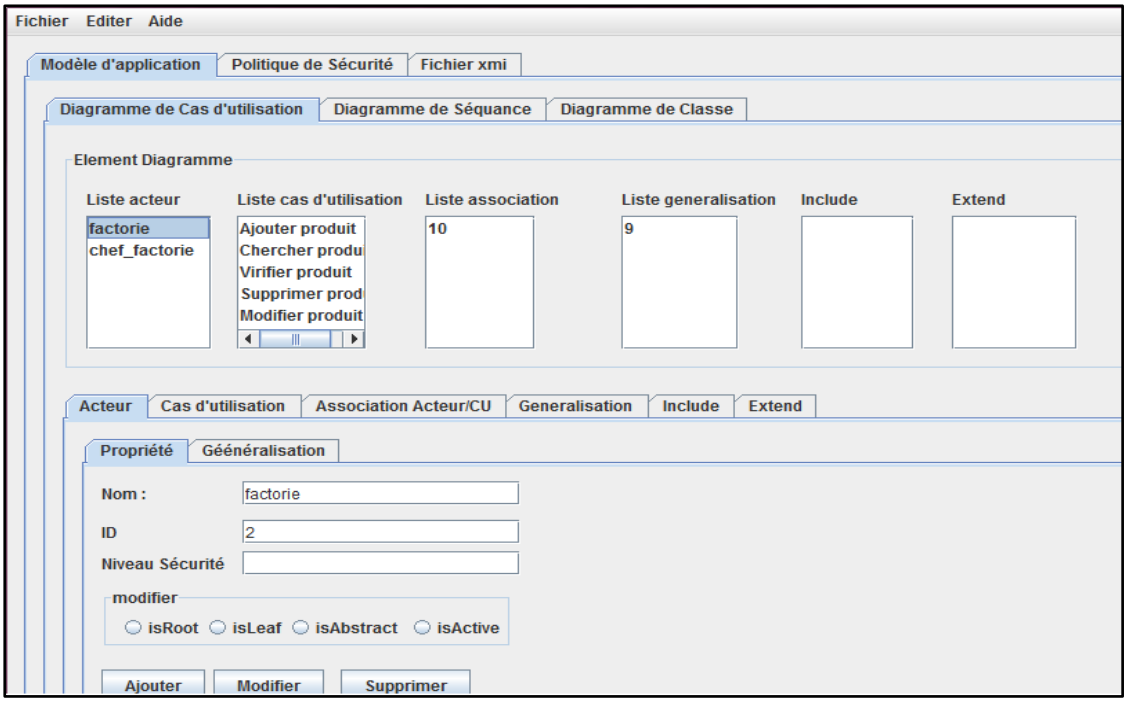


Fig.III.48. Interface de diagramme de cas d'utilisation.

- **Interface de diagramme de classe :**

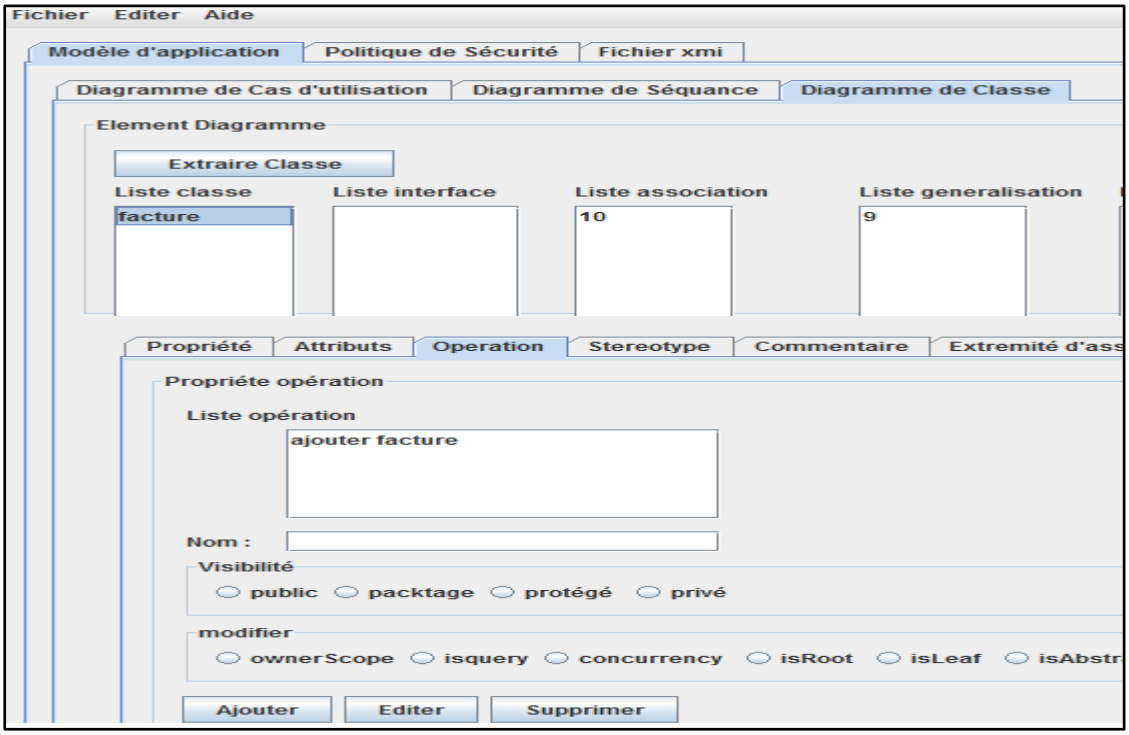


Fig.III.49. Interface de diagramme de classe.

- **Interface de diagramme de séquence :**

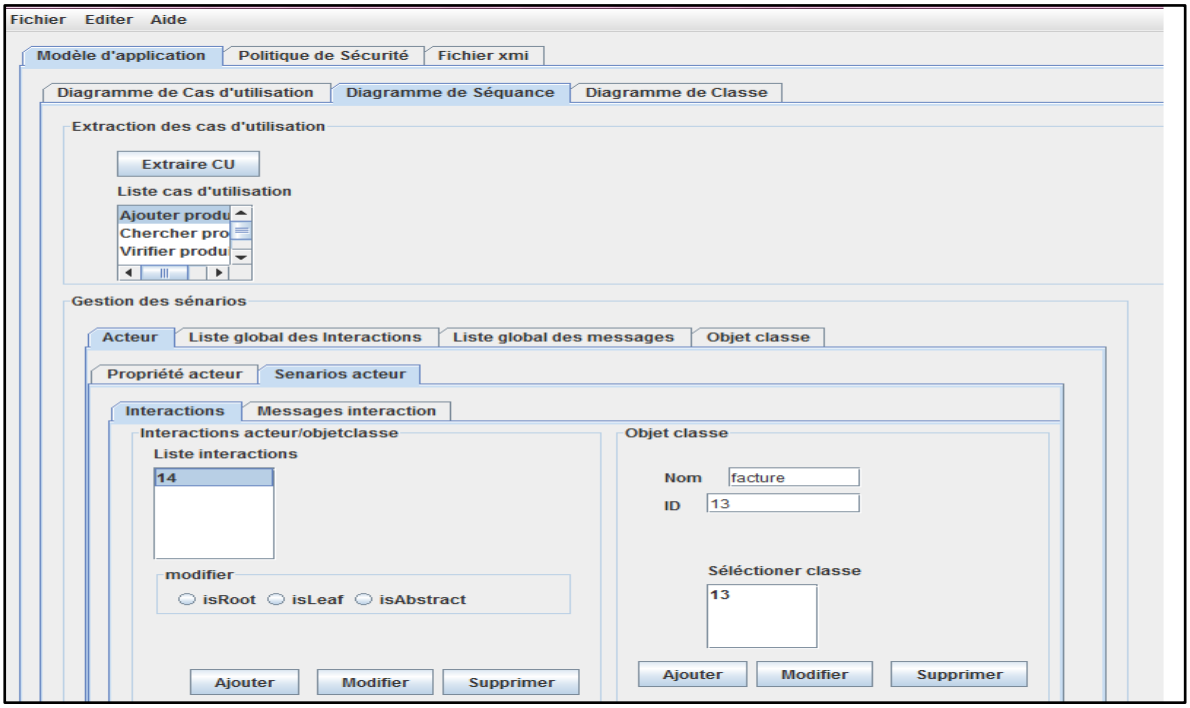


Fig.III.50. Interface de diagramme de séquence.

- **Interface de diagramme de séquence (ajouter message) :**

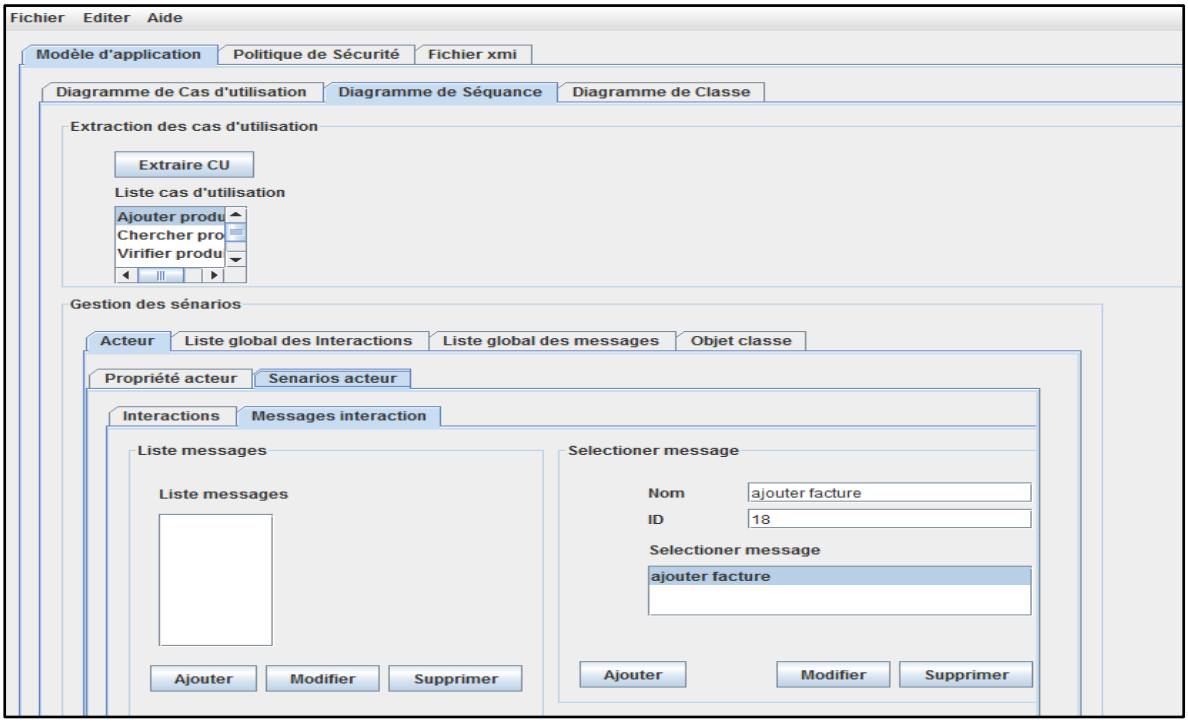


Fig.III.51. Interface de diagramme de séquence(Ajouter message).

- **Interface de diagramme de séquence (afficher les messages) :**

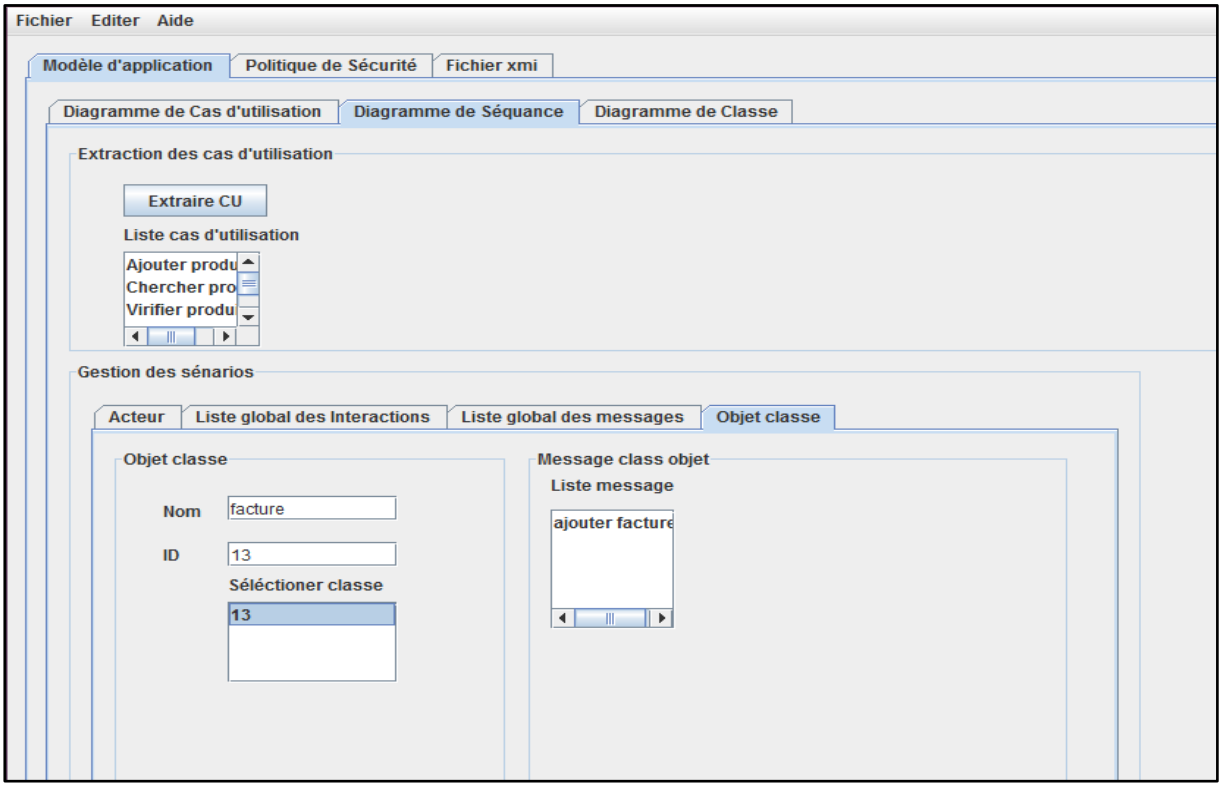


Fig.III.52. Interface de diagramme de séquence (afficher les messages).

- **Interface politique de sécurité (Configuration niveau sécurité):**

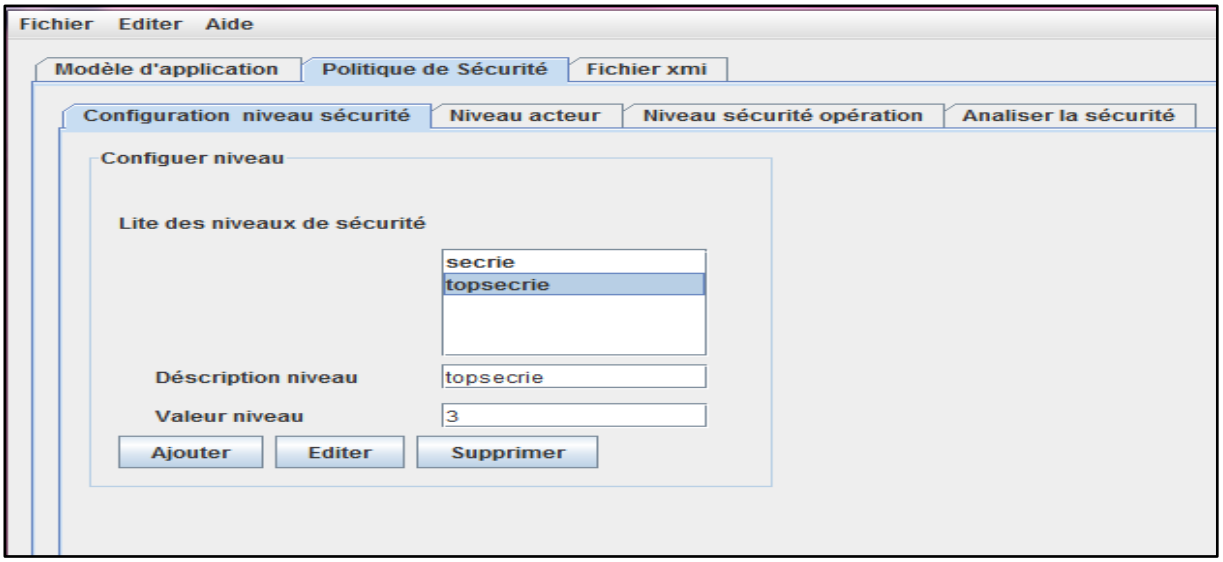


Fig.III.53. Interface politique de sécurité (Configuration niveau sécurité).

- **Interface politique de sécurité (Configuration niveau sécurité acteur):**

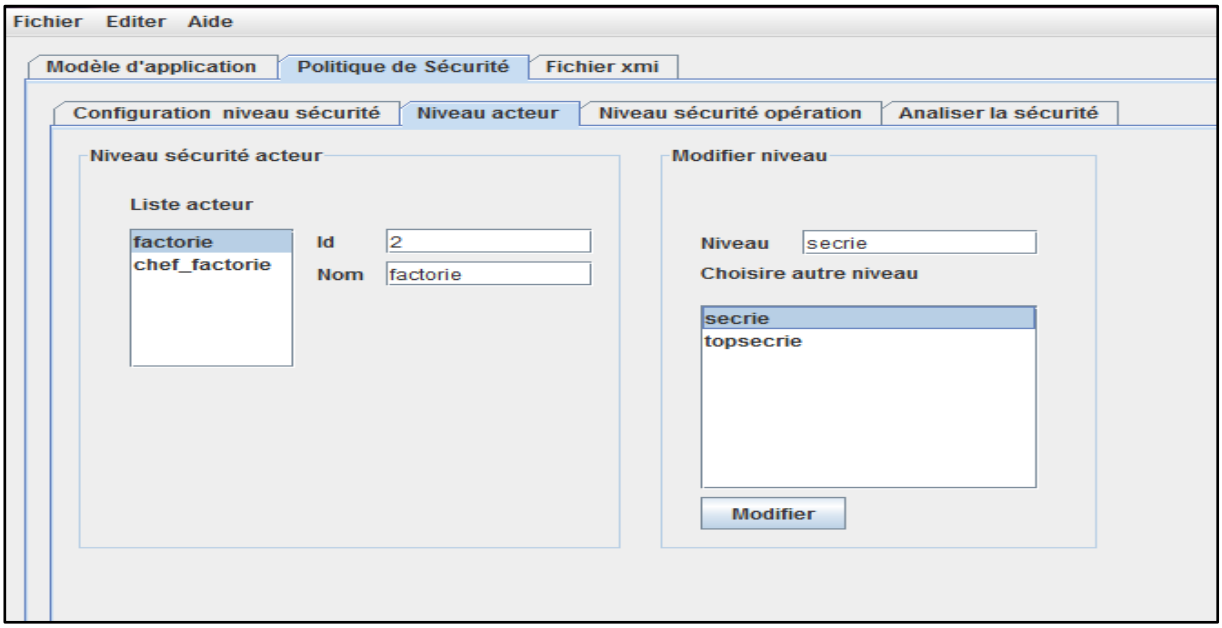


Fig.III.54. Interface politique de sécurité (Configuration niveau sécurité acteur).

- **Interface politique de sécurité (Configuration niveau sécurité opération):**

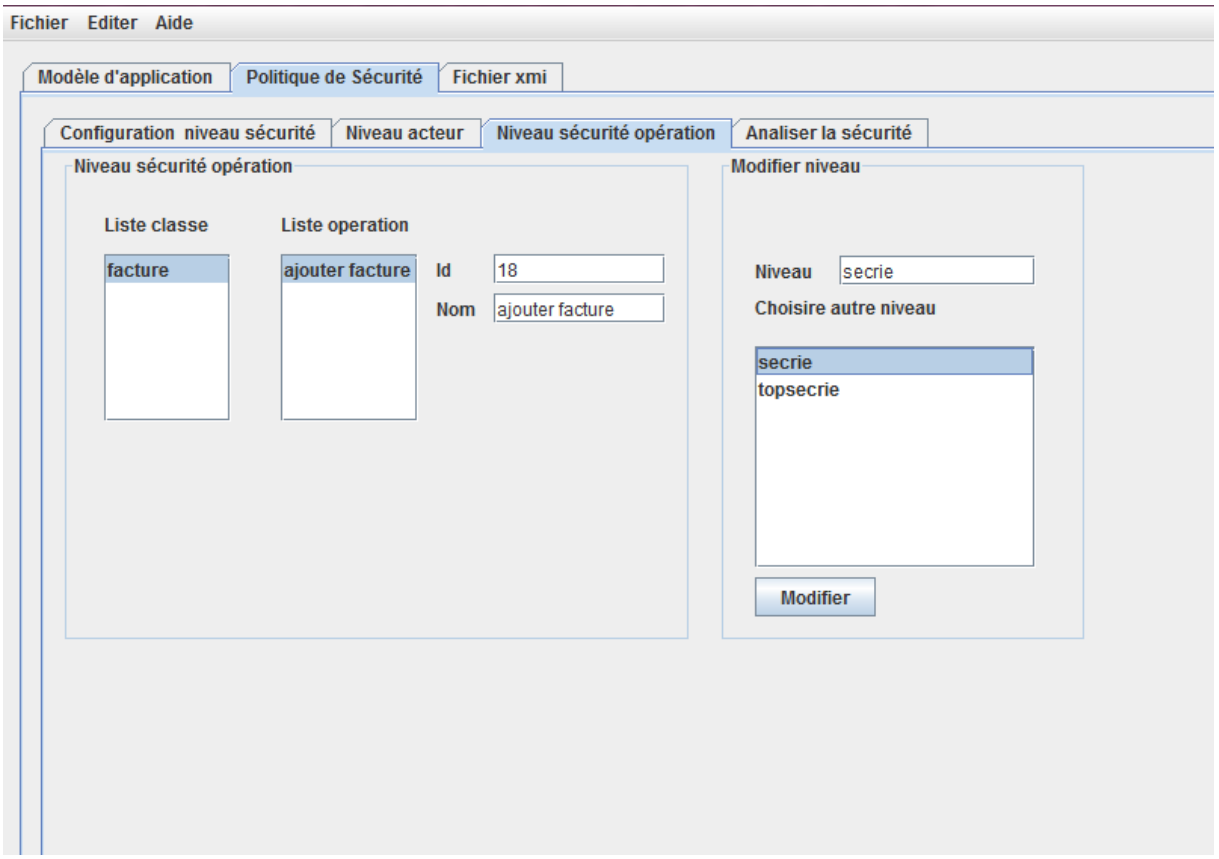


Fig.III.55. Interface politique de sécurité (Analyser la sécurité).

- **Interface politique de sécurité (Analyser la sécurité):**

Fig.III.56. Interface politique de sécurité (Analyser la sécurité).

- **Interface gestion de fichier XMI**

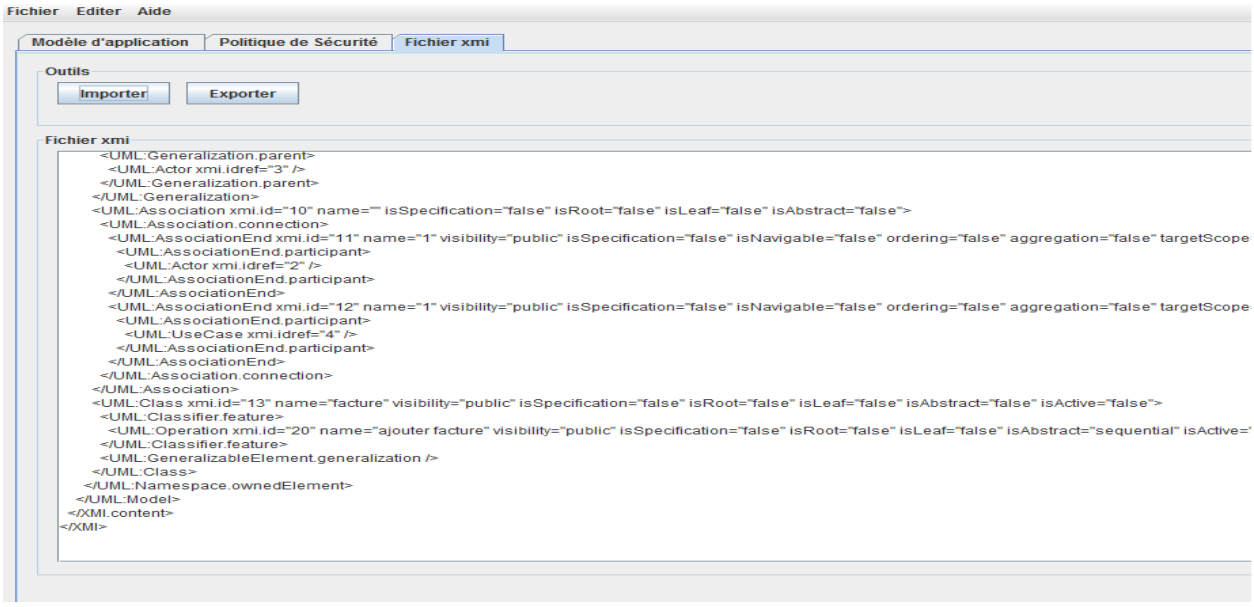


Fig.III.57. Interface gestion de fichier XMI.

V. Conclusion

Dans ce chapitre nous avons mentionné avec détail les outils que nous avons utilisés pour réaliser notre application, on a commencé par une présentation du langage JAVA, suivi par la présentation de NetBeans qui est un environnement de développement intégré (EDI), ensuite nous avons présenté XML plus précisément XMI, l'API JDOM pour représenter et manipuler un document XML de manière intuitive pour un développeur Java et enfin les principales utilisations et fenêtres de l'application.

Conclusion générale

Conclusion générale et perspective

Notre projet permet de renforcé la sécurité des systèmes d'information modélisé en UML en utilisant les modèles de contrôle d'accès pour cela nous avons commencé par une étude sur les contrôle d'accès et UML.

Le contrôle d'accès permet de contrôler les accès au système, il est classé en trois classes DAC, MAC et RBAC toute fois notre choix c'est porté le MAC pour son efficacité dans les systèmes à forte composante hiérarchique

UML est un langage de modélisation unifié qui permet de modéliser des applications selon ces différentes vue, toute fois UML n'est pas une méthode, d'où la nécessité d'une méthodologie qui permet la modélisation fiable du système.

L'utilisation des modèle de contrôle d'accès nous a permis de palier les problèmes liés aux contrôle d'accès et cela des les premières phases d'analyse d'une application, ce qui rend l'application plus fiable et plus résistantes aux menaces des hackers.

Enfin, nous terminons notre mémoire en présentant plusieurs perspectives que nus exposons dans ce qui suit :

1. Étendre l'étude sur les autres diagrammes d'analyse.
2. Étendre l'étude sur les diagrammes de conception.
3. Contrôler les accès avec les autres modèles de sécurité.
4. Contrôler les accès entre les systèmes.

Ce projet nous a permis de:

- Comprendre le méta-modèle UML et les modèles de contrôle d'accès (DAC, MAC et RBAC) et l'utilisation de la modélisation UML.
- Connaitre le langage XML et langage de programmation JAVA.
- Comprendre toutes les phases de la création d'une application.

Bibliographies

[1] J. Gabay et D.Gabay,UML2 Analyse et Conception, Groupe Dunod, ISBN : 978-2-10-053567-5, 2008.

[2] B.Combemale, Ingénierie Dirigée par les modèles (IDM), Institut de Recherche en Informatique de Toulouse(UMR CNRS 5505), 29 Mar 2009.

[3] C.Hardebolle, Composition de modèles pour la modélisation multi-paradigme du comportement des systèmes, Thèse Doctorat, l'université de Paris-Sud XI UFR Scientifique d'Orsay Mention Informatique, 28 Novembre 2008.

[4] T.Ziadi, Manipulation de Lignes de Produits en UML, Thèse Doctorat, l'université de Rennes 1 Mention Informatique, 13 Décembre 2004.

[5] X. Blanc,MDA en Action Ingénierie logicielle guidée par les modèles, Groupe Eyrolles, ISBN : 2-2012-11539-3, 2005.

[7] B. Mouna, Une approche basée transformation de graphes pour la génération de modèles de réseaux de Petri analysables à partir de diagrammes UML, Thèse Doctorat, l'université de CONSTANTINE 2, 24/ 04 /2013.

[8] P. A. Muller et N.Gaertner, Modélisation objet avec UML, Deuxième l'édition Groupe Eyrolles, ISBN : 2-212-11397-8, 2004.

[9]Y. Deswarte, La sécurité des systèmes d'information et de communication, en Sécurité des réseaux et des systèmes répartis, Hermès, ISBN 2-7462-0770-2, octobre 2003.

[10]ITSEC, Information Technology Security Evaluation Criteria, v 1.2, ISBN 92-826-3005-6, Office des publications officielles des Communautés Européennes, Luxembourg, 1991.

[11] A. Abou El Kalam, Modèle et politique de sécurité pour le domaine de la santé et des affaires sociales, Thèse de Doctorat, Laboratoire d' Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique, Toulouse, décembre 2003.

[12] A. Haddad, Modélisation et vérification de politique de sécurité, Formation Européenne de 3ème Cycle en Systèmes d'Information: MATIS Université de Joseph Fourier, Genève, 2005.

[13]S.Medjdoub. Modèle de contrôle d'accès pour XML :"Application à la protection des données personnelles". Thèse de doctorat, Université de Versailles-Saint Quentin en Yvelines,décembre 2005.

Bibliographies

[14] S.Boulares, Validation des politiques de sécurité par rapport aux modèles de contrôle d'accès ,Thése du maîtreès sciences (M.sc.), Université du Québec en Outaouais, Aout 2010.

[15]A. Michael Harrison, W. L.Ruzzo , Berkeley Jeffrey D. Ullman, Princeton University Protection in Operating Systems,University of California, Copyright Q 1976.

[16]K.Bouriche, Gestion de l'incertitude et codage des politiques de sécurité dans les systèmes decontrôle d'accès, Thèse Doctorat, Université d'Artois et de l'Université Sidi Mohammed Ben Abdellah Spécialité Informatique.

[17] D. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli, A Proposed Standard for Role Based Access Control, ACM Transactions on Information and System Security, vol. 4, no. 3, August 2001.

[18] D. Basin, J.Doser, T.Lodderstedt, Model Driven Security: from UML Models to Access Control Infrastructures, Information Security Group, ETH Zurich Interactive Objects Software GmbH Freiburg, Vol 15,N°1, September 2006.

[19] D. Gros, Contrôle d'accès mandataire pour Windows 7, Thèse Doctorat, Université France, 7 Juin 2012.

[20] Abounasr meryrm et boujadisoukaina, rapport transformation des fichiers XMI en fichier SVG avec JAVA et ATL, universite Hassan II-mohammedia, 2014/2015.

[21] OMG Document Number: formal/2013-06-03 Standard document URL: http://www.omg.org/spec/XMI/2.4.1.

[22] http://www.jdom.org/