MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ ABDELHAMID IBN BADIS - MOSTAGANEM

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

MÉMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : **Ingénierie des Systèmes d'Information**

THÈME :

Conception d'un système de détection de faille dans les systèmes modélisés en UML (Diagramme de composant et diagramme de déploiement étendu)

Etudiante : BAGHDADI Sihem

Encadrant : KHELIFA Noreddine

Année Universitaire 2015/2016

Résumé

Dans le domaine de la sécurité informatique une faille est une faiblesse dans un système informatique permettant à un attaquant de porter atteinte à son intégrité.

Afin de garantir la sécurité des applications, nous proposons un logiciel capable de détecter les vulnérabilités des systèmes et cela dés la phase de modélisation, et plus particulièrement dans la partie spécification de l'architecture physique de l'application.

En UML, l'architecture physique de l'application est modélisée par les diagrammes de composant (fichiers de données, fichiers sources …) et les diagrammes de déploiement (matériel informatique..).

Dans notre application, à ces deux diagrammes (composant et déploiement) une politique de contrôle d'accès est ajoutée, cette politique permet d'empêcher qu'un utilisateur puisse exécuter des opérations qui ne lui sont pas permises.

Pour arriver à nos fins, des algorithmes sont implémentés pour vérifier l'existence de failles qui peuvent nuire au bon fonctionnement de l'application, ces algorithmes permettent de détecter les goulots d'étranglement, de prévenir les problèmes de congestion, d'alerter sur l'existence de points d'articulation et repérer les accès non autorisés.

Mots clés :

UML, méta-modèle, faille, vulnérabilité.

Dédicaces

Je dédie ce travail à :

Ma très chère mère « Amina » qui m'a toujours épaulé et motivé et que ses conseils me suivront toujours où je serai.

Je n'aurais pu réussir mes études sans elle à qui, quoi que je dirai ne suffira pas pour exprimer ma gratitude et ma reconnaissance pour son soutien, ses encouragements continus pour aller sans cesse en avant, et que je rassure que je serai à la hauteur de ses espérances.

Qu'elle trouve dans ce travail l'expression de mon grand amour et ma grande gratitude, et que dieu lui préserve bonne santé et longue vie.

Ma sœur « Sonia » à qui je souhaite un brillant long parcours dans son travail, ainsi qu'à son mari « Fayçal » lui assurant que je n'oublierai jamais ses services.

Mes frères « Amine,Islem » ,mes cousins et cousines et particulièrement « Souhila » et à toute la famille Baghdadi, Dellal.

Je n'y manquerais pas aussi de dédier ce travail à mes copines les plus intimes.

A ceux qui m'ont encouragée et soutenue dans mes moments les plus durs.

A tous les étudiants du département d'Informatique et spécialement la promotion Master ISI 2015/2016.

Remerciement

Avec l'aide d'Allah, est achevé le présent travail

Ce mémoire est le résultat d'un travail de recherche de près de quatre mois. En préambule, je veux adresser tous mes remerciements aux personnes avec lesquelles j'ai pu échanger et qui m'ont aidé pour la rédaction de ce mémoire.

Mes premières pensées vont à remercier la plus belle personne dans ma vie... A une femme qui m'a toute donnée...A ma mère...

Merci Maman de m'avoir donné tant d'amour et de tendresse, et merci de m'avoir toujours poussé dans mes intérêts.

Mes remerciements les plus vifs à mon encadrant Mr KHELIFA Noreddine*, qui a su me guider et m'aider dansce travail avec beaucoup de tact et de gentillesse et qui ma permis de découvrir un domaine très intéressant. Qu'il trouve ici mon estime et mon profond respect.*

Je tiens également à remercier toutes les personnes qui ont participé, à titre professionnel ou personnel à la réalisation de ce travail en particulier Mr BEnamer.

Mes remerciements iront également vers tous ceux qui ont accepté avec bienveillance de participer au jury de ce mémoire.

A toutes mes copines « Yasmine,Amina,Souad,Ilhem,Narimen » qui m'ont accompagné tout au long de mes cinq ans de maîtrise.

Je suis particulièrement redevable à ma copine Aberkane Sabrina, je la remercie pour sa patience, son aide et son encouragement.

J'ai également une pensée amicale pour « Walid », dont je garderai l'exemple ainsi merci pour ton soutien moral.

Je remercie chaleureusement monsieur bekkeouche Mohamed.

Je remercie enfin toutes les personnes intéressées par mon travail, en espérant qu'elles puissent trouver dans mon rapport des explications utiles pour leurs propres travaux

Sommaire

Liste des figure

Liste des tableaux

Liste des Abréviations

UML : Unified Modeling Language ou langage de modélisation unifié.

DAC: Discretionary access control ou Contrôle d'accès discrétionnaire

MAC: Mandatory access control ou contrôle d'accès obligatoire

RBAC: Role-Based Access Control ou contrôle d'accès à base de rôles

UP7: Unfied process ou processus unifié

RUP: rational unified process ou processus unifié.

XP :eXtreme Programming.

–

Introduction générale

De nos jours les attaques causent de plus en plus de dégâts financiers et perte de temps, beaucoup de ces attaques viennent du faite que les pirates exploitent des failles existantes dans l'application.

Ces failles peuvent être causées par la négligence des utilisateurs, la mauvaise administration de l'application, ou par l'architecture physique de l'application elle-même qui ne se conforme pas avec la politique de sécurité,…..

Dans la phase de modélisation, La spécification de l'architecture physique de l'application est une pratique très courante elle permet de donner une vue sur comment le système sera déployé en matière de composant et de nœud et leurs relations.

En UML cette spécification est modélisée par deux diagrammes : le diagramme de composant et le diagramme de déploiement qui sont des diagrammes proches du monde réel de l'application (dépendance des fichiers, interconnexions des machines…) d'où notre intérêt d'intégrer des mécanismes qui permettent de déceler les problèmes de sécurité a ce niveau.

Notre projet a été divisé en quatre chapitres :

Chapitre 1 : Le langage de modélisation UML

On présente UML (unified modeling language), ses diagrammes et ses mécanismes d'extension. Ensuite, nous faisons une description détaillée sur la méta-modélisation, particulièrement méta-modèle UML, conformément à la définition standardisée par l'OMG.

Chapitre 2 : La sécurité

On définit la sécurité, les politiques de sécurité et les contrôles d'accès ainsi que leurs différents types (DAC, MAC et RBAC).

Chapitre 3 : Analyse et Conception (partie théorique)

Nous proposons l'analyse et la conception de notre projet. Nous définissons tout d'abord, la vue de haut niveau de l'application ensuite nous proposons les

algorithmes qui détectent les problèmes liés à la sécurité et en fin nous modélisons notre application.

Chapitre 4 : L'implémentation

Dans ce chapitre nous définissons le langage java et l'outil ''NETBEANS'' avec le quel nous avons développé notre application et nous finissons par un testes appliqué a notre application.

Chapitre 1 : UML et méta-modèle.

I.1. Introduction

L'utilisation du langage de modélisation unifié UML dans l'étape de modélisation est devenue une pratique très courante, vu qu'il offre une multitude de diagrammes qui assurent de différentes visions sur le système (fonctionnel et structurel), ces diagrammes associés à une méthode peuvent assurer aussi une transition entre le cahier de charge et le code source, et en plus leur structure est définie par le méta modèle.

Dans ce chapitre nous allons commencer par définir l'UML et ses diagrammes puis nous expliquerons le méta modèle et enfin nous proposerons deux méta-modèles qui concernent le sujet de notre projet.

I.2. Définition UML

UML (UnifiedModelingLanguage ou «langage de modélisation unifié») est un langage de modélisation visuelle universelle qui est utilisé pour spécifier, visualiser, construire et documenter les artefacts d'un système logiciel. Il modélise un système à partir de ses besoins exprimés dans un cahier de charge jusqu'à la phase de création des éléments du code source. Il est utilisé pour comprendre, concevoir, feuilleter, configurer, maintenir (entretenir) et contrôler des informations sur de tels systèmes [1,2]

I.3. Version d'UML

Les premières versions d'UML ont été créées par GradyBooch (le créateur de méthode de Booch), Ivar Jacobson (le Génie logiciel Orienté objet, OOSE) et Jim Rumbaugh (la Technique modelant l'Objet, OMT).

Les mises à jour d'UML sont gérées par le Groupe de Gestion d'Objet (OMG), qui a publié la dernière version UML 2.5 en juin 2015, cette version à connu une amélioration sur le méta modèle (des noms changés,…), les notations (les membres hérités sont désignés parle signe"^"…) et le profile (StandardProfileL2 etL3StandardProfileont été fusionnés en un seul…) par rapport à la version précédente UML 2.4 [3].

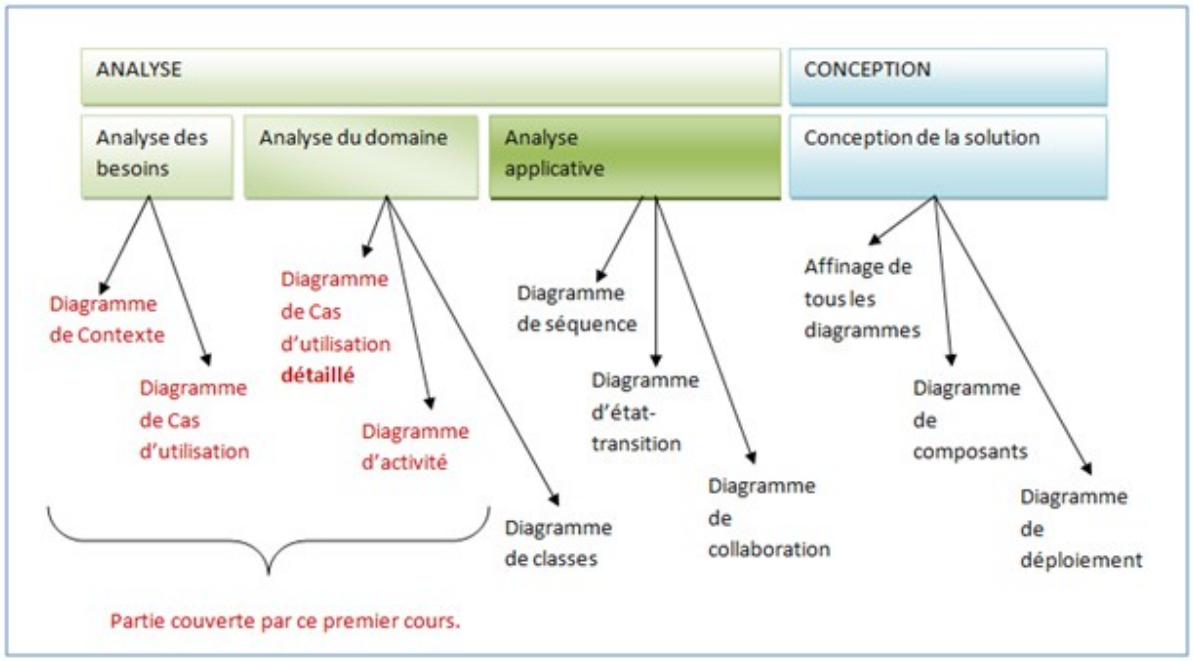

Figure I.1. Fonctionnement UML.

I.4. 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 [3].

- **Les diagrammes structurels (Statique) :** montrent la structure statique du système et ses parties sur différentes abstractions, ce type de diagramme permet de montrer des éléments et leurs relations.
- **Les diagrammes de comportement (Dynamique)** : montrent le comportement dynamique des objets dans un système, qui peut être décrit sous forme d'une série de changements opérés dans le système au fil du temps [3].

La figure I.2 montre les différents diagrammes existant dans la version 2.5.UML définit deux sortes majeures de diagramme d'UML : Diagrammes de structure et diagrammes de comportement.

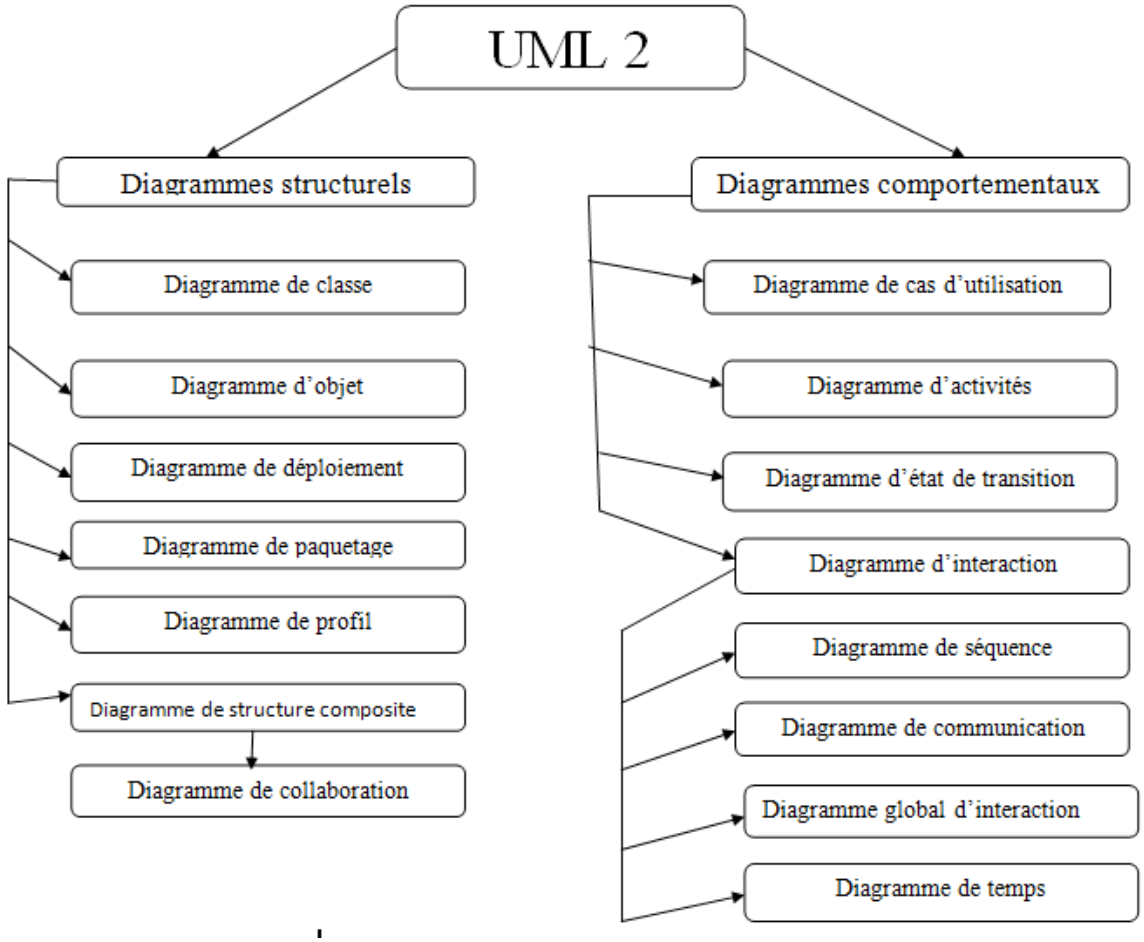

Figure I.2. Les 14 diagrammes d'UML.

I.4.1. Diagrammes structurels (statiques)

- **Diagramme de classe :** Une classe est représentée par un rectangle divisé en trois compartiments : le compartiment du nom, le compartiment des attributs et le compartiment des opérations. Dans la phase d'analyse, ce diagramme représente les entités manipulées par les utilisateurs, dans la phase de conception, il représente la structure objet d'un développement orienté objet

Les Relations entre classes

Un diagramme de classes montre les classes d'un système mais également les relations entre eux, on compte quatre principales catégories de relations entre classes:

1. Les associations
2. Les généralisations (héritage)
3. Les dépendances
4. Les raffinements [4].

- **Diagramme d'objet :** Ensemble d'objet respectant les contraintes du diagramme de classe, il sert à représenter les instances de classes, il montre des objets et des liens entre ces objets.
- **Diagramme de package :** Regroupement d'éléments (classe, cas d'utilisation, interface), il permet de décomposer le système en catégories ou parties plus facilement à observer appelée « package » cela permet également d'indiquer les acteurs qui interviennent dans chacun des packages.
- **Diagramme de composant :** Un composant est une entité physique qui réside sur un nœud matériel et qui peut être exécuté ou participé à l'exécution, il représente les composants physiques d'une application, il montre la structure complexe avec leur interface fournie et requise il est très utilisé dans le domaine de recherche [1, 5].
- **Diagramme de structure composite :** Un ensemble d'éléments interconnectés collaborant dans un but commun lors de l'exécution d'une tache, il montre l'organisation interne d'un élément statique complexe.
- **Diagramme de déploiement :** Il montre la configuration physique des différents matériels qui participent à l'exécution du système ainsi que les artefacts « une entité physique représenté par un rectangle contenant le mot-clé « artefact » qu'il supporte. Il comprend: nœuds, connexions, composants et objets [1,3].

Notion d'artefact

- Un artefact est un classificateur qui représente une certaine entité physique.
- Une information qui est utilisée ou est produite par un processus de développement logiciel, ou par le déploiement et l'opération d'un système.
- Un artefact est présenté par un rectangle de classe ordinaire avec le mot-clé "l'artefact"[1,2].

I.4.2. Diagrammes comportementaux (dynamiques)

- **Diagramme d'activité:** c'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 des cas d'utilisation :** Il capture le comportement d'un système, d'un sous-système d'une classe ou d'un composant tel qu'un utilisateur peut les visualiser, il permet d'exprimer les besoins des utilisateurs, il contient les éléments suivants: acteur (rôle joué par une personne externe), cas d'utilisation (fonctionnalité visible de l'extérieur
- **Diagramme de séquence :** Il représente une séquence d'interaction temporelle entre des objets, ces interactions sont représentées sous forme de séquence verticale des messages.

- **Diagramme de collaboration :** C'est un diagramme de classes qui contient des rôles de classificateur et des rôles d'association, c'est une collection des objets qui interagissent pour mettre en œuvre le comportement dans un contexte.

- **Diagramme de communication :** Il montre la communication entre objets, dans un plan au sein d'une interaction. On donne le séquencèrent de messages par numérotation ordonné sur le plan.

- **Diagramme d'état de transitions :** Il montre les différents états possibles des objets d'une classe. Les objets répondent à des événements externes. Ce diagramme peut aussi exprimer du parallélisme et fournit une forte capacité d'abstraction pour des comportements associés aux états**.**

- **Diagramme de profile :** Les diagrammes de profiles sont des diagrammes de classes simplifiées, représentant les notions de profiles et de classes. Il contient le profil, la méta classe, le stéréotype, l'extension, la référence.

- **Diagramme globale :** Il met le focus sur le flot général de contrôle dans lequel chaque nœud représente un diagramme d'interaction. Il s'agit d'un cas spécial de diagramme d'activité.

- **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 [3].

I.5. 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'est à dire le langage de modélisation. Un langage de modélisation est défini par une syntaxe abstraite, syntaxe concrète et sémantique.
La syntaxe abstraite définit les concepts de base du langage [6].

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 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.

I.6. 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 Figure I.3).

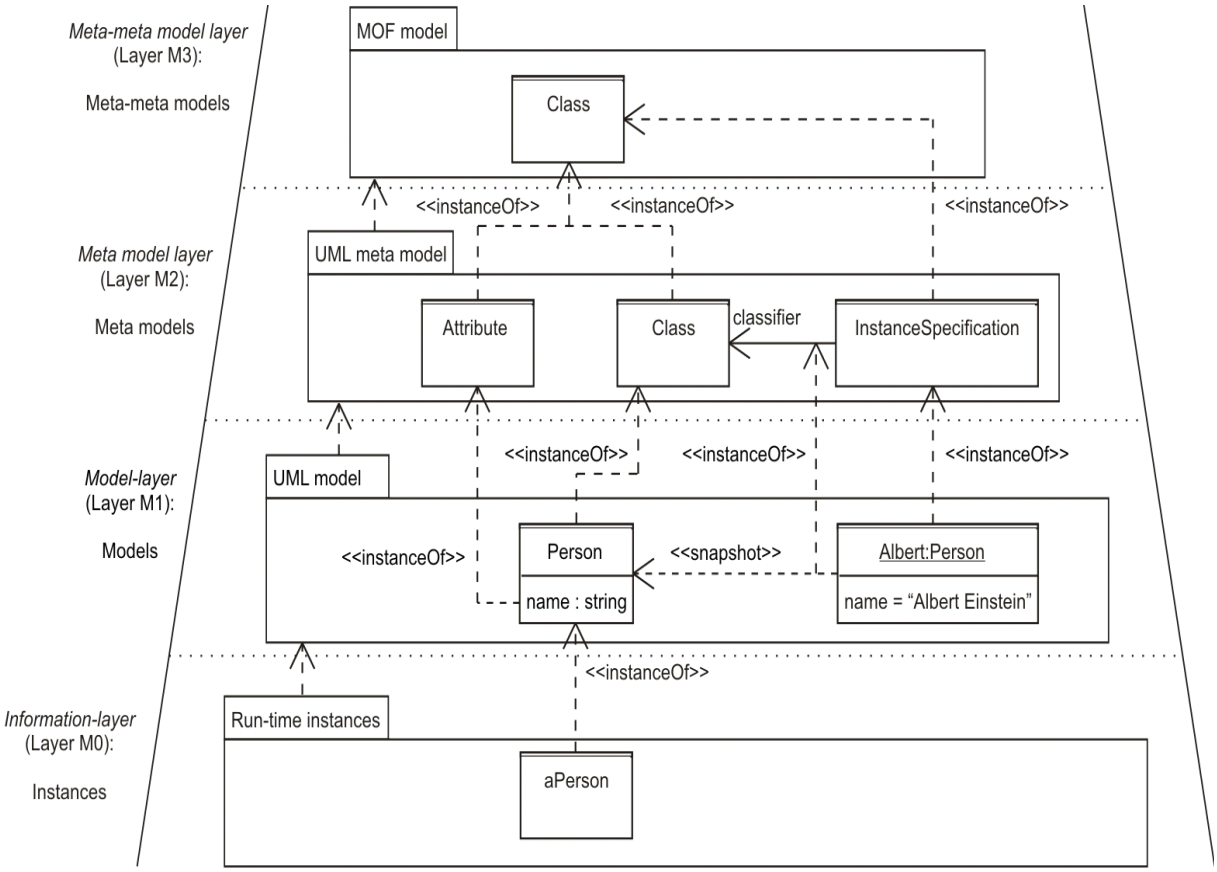

Figure I.3. L'architecture d'UML à quatre niveaux de l'OMG.

- **M3 (méta-méta-modèle) :** Cette couche définit un langage de haut niveau pour spécifier le méta modèle. Elle est instance d'elle même. Le méta modèle d'UML (niveau M2) en est une instance particulière.
- **M2 (méta-modèle) :** C'est la couche qui définit les éléments syntaxiques et sémantiques des modèles du niveau M1. Les concepts UML, "Class", "Attribut", "Instance", etc., sont définis à ce niveau.
- **M1 (modèle) :** C'est la couche qui héberge les modèles UML développés pour spécifier un domaine d'application donné. Elle décrit la structure de la couche 0.
- **(M0) (objet)** : c'est la couche qui correspond au niveau des objets réels, elle est composé des informations que l'on souhaite modéliser. Cette couche est souvent considérée comme étant le monde réel.

Le méta-modèle d'UML est décrit en utilisant une partie de la notation d'UML [6,7].
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 [6]

I.6.1. Le méta-modèle de diagramme de composant

Le méta-modèle de ce diagramme (figure I.4) contient des composants sous forme de classe. Une classe peut contenir un attribut. Ce dernier a un nom et une visibilité (public, privé, protégé), et un type qui peut être soit string, integer ou boolean. Le composant a un nom.

Le composant peut utiliser d'autre composant comme il peut être utilisé à son tour. Le port est une propriété d'un composant qui a une interface comme type, une partie est un attribut d'un composant dont le type est un composant, la dépendance indique qu'une interface requise sur un composant peut être satisfaite par une autre interface fournie. La généralisation indique qu'un composant hérite d'un autre composant.

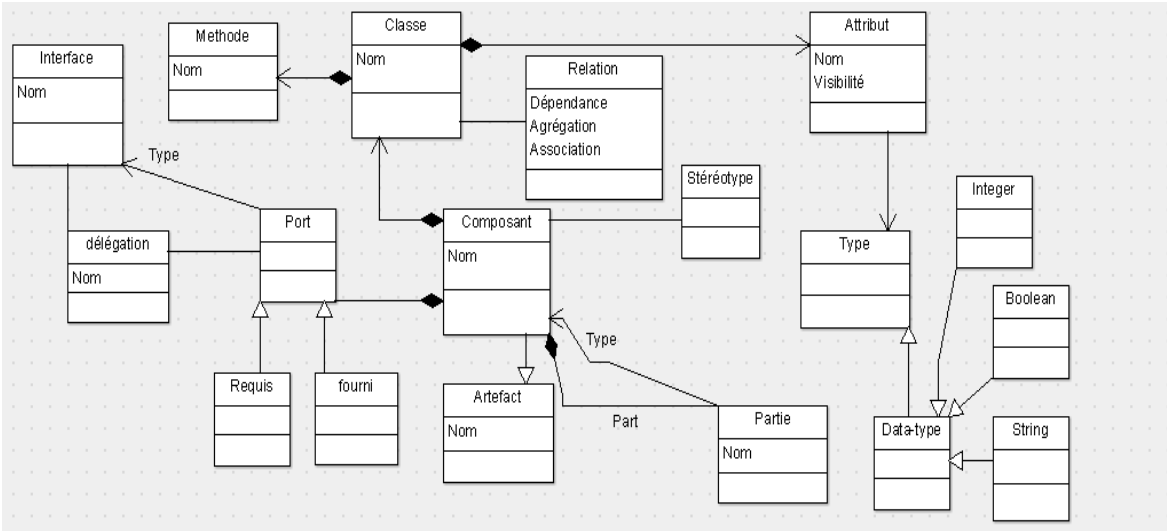

Figure I. 4. Méta-modèle du diagramme de composant.

I.6.2. Méta-modèle du diagramme de déploiement

Le méta-modèle de ce diagramme contient (figure I.5) des nœuds qui ont un nom et des attributs. Un nœud renferme des composants qui ont un nom et des associations comme lien de communication. La dépendance est une relation stéréotypée qui peut être soit deploy (artefact et nœud) ou manifest (artefact et composant), l'artefact est utilisé pour définir un fichier, un programme, une bibliothèque ou une base de données construite ou modifiée dans un projet.

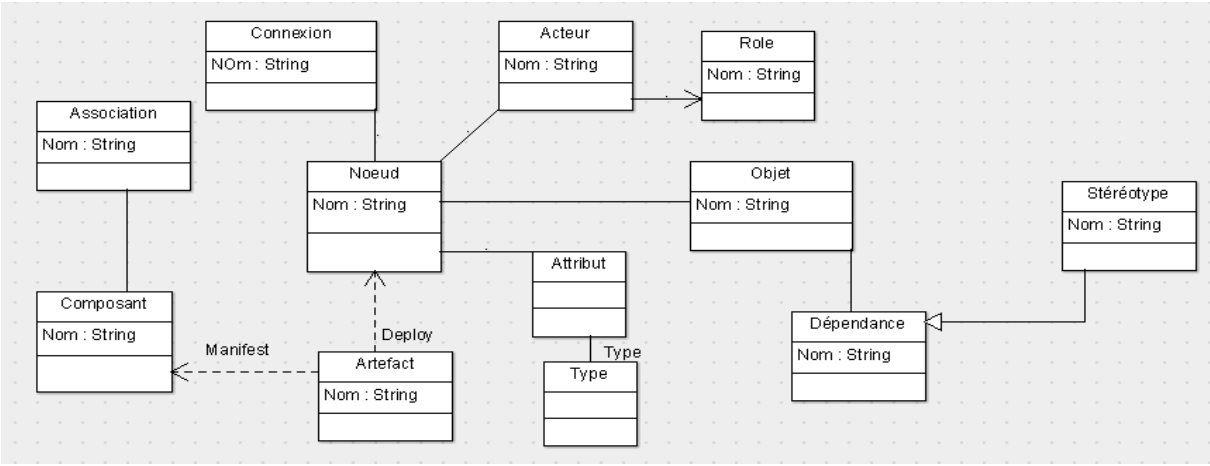

Figure I.5. Méta-modèle du diagramme de déploiement.

I.7. Les mécanismes d'extension

Pour répondre à ces besoins particuliers, l'OMG a introduit depuis la versionUML1.3 des mécanismes dits d'extension permettant de spécialiser et d'adapter UML à des domaines, des plates-formes ou des méthodes spécifiques. Les mécanismes d'extension introduits depuis la version UML1.3 sont les Tagged Values, les Stéréotypes, et les Contraintes. La notion de Profil est introduite comme un moyen pour structurer et regrouper les trois mécanismes précédents.

- **Les stéréotypes**

Le stéréotype est le mécanisme de base pour l'extension d'UML. Il définit comment une méta-classe particulière du méta-modèle d'UML peut être étendue pour permettre l'utilisation d'une terminologie ou d'une notation spécifique à un domaine ou une plate-forme particulière. Un stéréotype est lié par un lien d'extension à une méta-classe particulière du méta-modèle [8].

- **Les Profils**

Un profil, par définition, étend un méta-modèle de référence ou un autre profil. Le méta-modèle de référence peut être le méta-modèle d'UML ou un autre méta-modèle basé sur la librairie d'infrastructure. L'appellation profil UML est utilisée pour désigner un profil dont le méta-modèle de référence est le méta-modèle UML [8]

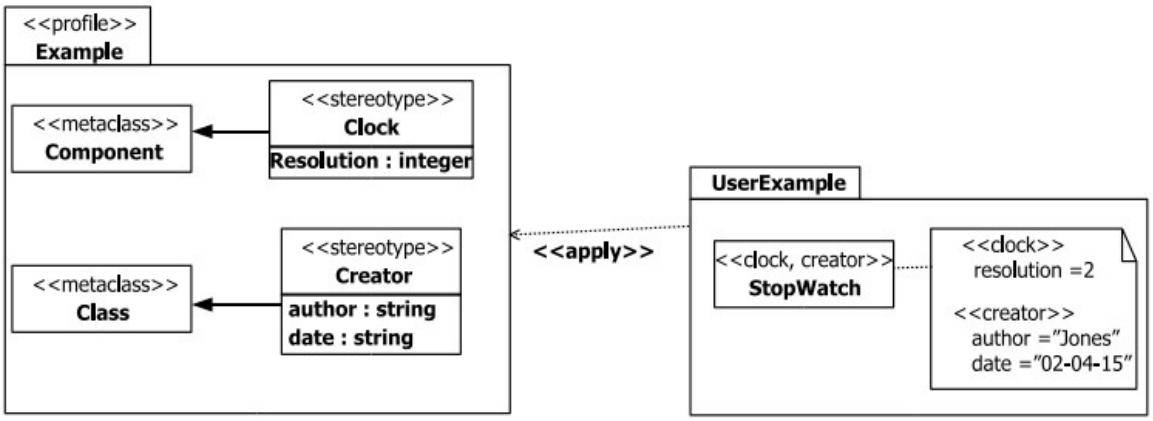

Figure I.6. Exemple de spécification d'un profil UML2.0 et son utilisation

La Figure I.6 montre un exemple du profil UML appelé Example regroupant lesdeux stéréotypes Clock et Creator. Les profils dans UML2.0 sont notés comme despaquetagesUML avec le stéréotype <<profile>>. Le paquetage UserExampleestun exemple de modèle utilisateur basé sur le profil Example. La classe StopWatchestdéfinie avec les deux stéréotypes de profils et les tagged values associées aux stéréotypessont définis comme des notes UML (voir la Figure I.4) [8].

I.8. Conclusion

Nous avons défini dans ce chapitre d'une manière assez détaillée la notation UML qui est un langage de modélisation unifié, son évolution, ses diagrammes et ses mécanismes d'extension, une description détaillée sur la méta-modèle UML qui est adoptée par l'OMG est connue comme une hiérarchie à quatre niveaux.

Ainsi, et les deux diagrammes qui concernent notre projet de fin d'étude (Composant et déploiement).

Dans ce qui va suivre nous allons mettre l'accon sur la sécurité et ces différentes politiques et le modèle adopté par notre projet.

II.1. Introduction

La sécurité informatique revêt une importance particulière qui grandit avec le développement fulgurant des technologies de l'information et de la communication. Le contrôle d'accès qui représente une composante importante de la sécurité des systèmes d'information consiste à vérifier si un sujet demandant d'accéder à un objet possède les droits nécessaires pour le faire. Ainsi, la sécurité informatique consiste d'une manière générale à assurer que les ressources matérielles ou logicielles d'une organisation sont uniquement utilisées dans le cadre prévu.

Nous allons en premier lieu présenté dans ce chapitre la sécurité dans les systèmes informatique, abordé ensuite en second temps la politique de contrôle d'accès.

II.2. La sécurité informatique

La sécurité informatique est l'ensemble des moyens mis en œuvre pour minimiser la vulnérabilité d'un système contre des menaces accidentelles ou intentionnelles.

Différences entre accidents et malveillances:

- Sécurité ="Safety" :C'est la protection contre des événements accidentels imprévisibles.
- Sécurité = "Security" (Sureté) : C'est la prévention des dommages causés par les actions des attaquants.

Ainsi, la sécurité des systèmes d'information cherche à apporter une meilleure maîtrise des risques qui pèsent réellement sur l'entreprise et répondre à certains enjeux qu'on peut résumer en 3 lettres « DIC » (disponibilité, intégrité et confidentialité) [9,10].

II.2.1. Objectifs de sécurité

La sécurité informatique vise généralement cinq principaux objectifs :

- **La confidentialité** : empêcher une divulgation non autorisée de l'information.
- **L'intégrité :** empêcher une modification non autorisée.
- **L'authentification** : empêcher l'utilisation non- autorisée de ressources.
- **La disponibilité** : empêcher un déni non autorisé d'accès à l'information ou à des ressources.
- **La non répudiation** : garantir qu'une transaction ne peut être niée.
- **La responsabilité :** regroupe la disponibilité et l'intégrité de l'identité de la personne [11]

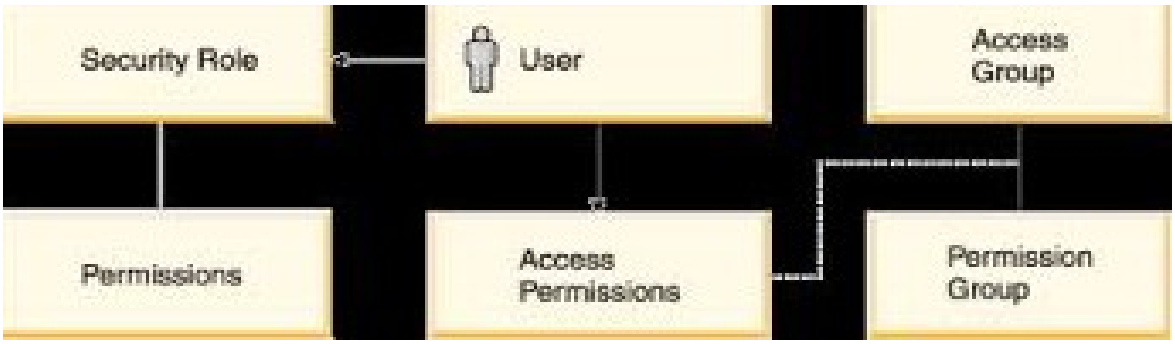

Figure II.1. Principe de la confidentialité

II.2.2. Concepts de sécurité

Le risque **(danger)** en terme de sécurité est généralement caractérisé par :

- **La menace (avertissement)** qui représente le type d'action susceptible de nuire dans l'absolu.
- **La vulnérabilité (fragilité ou faiblesse)** qui, appelée parfois faille, bug (défaut) ou brèche (trou) représente le niveau d'exposition face à la menace dans un contexte particulier.
- **La contre-mesure** qui est l'ensemble des actions mises en œuvre pour prévenir les menaces [10].

II.3.Contrôle d'accès

Le contrôle d'accès consiste à vérifier si une entité demandant d'accéder à une ressource a les droits nécessaires pour le faire. Un contrôle d'accès offre ainsi la possibilité d'accéder à des ressources physiques ou logiques. Il permet de gérer l'accessibilité des différents objets en suivant des règles bien précisent. Ainsi l'accès à des espaces sensibles peut être restreint à des utilisateurs identifiés.

- L'accès au système d'information exige une identification et une authentification préalable.
- L'utilisation de comptes partagés ou anonymes doit être évitée.
- Des mécanismes permettant de limiter les services, les données, les privilèges auxquels à accès l'utilisateur en fonction de son rôle dans l'organisation, doivent être mis en œuvre dans la mesure du possible.
- L'attribution et la modification des accès et privilèges d'un service doivent être validés par le propriétaire du service.
- Pour les services sensibles, un inventaire régulièrement mis à jour en sera dressé.
- Il importe de bien différencier les différents rôles et de n'attribuer que les privilèges nécessaires [12]

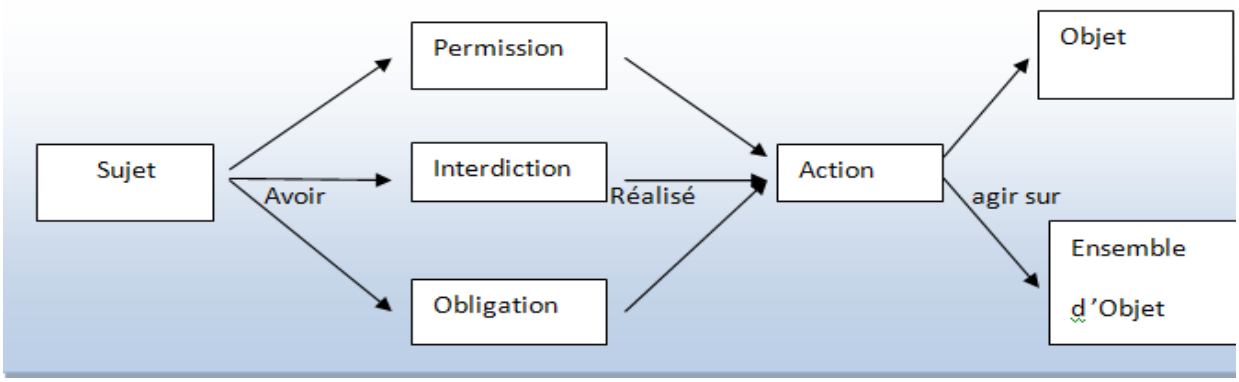

Figure II.2. Politique de contrôle d'accès

II.4. Politique de sécurité

Une politique de sécurité des systèmes d'information est un plan d'actions qui reflète la vision stratégique de la direction d'une organisation pour assurer des objectifs de sécurité tels que la confidentialité et l'intégrité. Ces derniers peuvent être représentés de façon claire et non ambiguë par des modèles de contrôle d'accès.

- **Les politiques discrétionnaires (DAC)** : Dans le cas d'une politique discrétionnaire, les droits d'accès à chaque information sont manipulés librement par le responsable de l'information (généralement le propriétaire), à sa discrétion.
- **Les politiques obligatoires (MAC)**: Une politique de sécurité d'autorisation obligatoire (ou MAC de l'anglais " Mandatory Access Control ") impose des règles d'autorisation incontournables qui s'ajoutent aux règles discrétionnaires.
- **Les politiques basées sur les rôles (RBAC) :** ces politiques sont les plus récentes, elles servent à décrire d'une manière plus expressive et plus puissante les fonctionnalités dans les organisations. Les droits d'accès sont accordés aux rôles, tâches dans l'organisation, ils sont attribués aux sujets en fonction des rôles qu'ils jouent [13].

II.4.1. Modèle de contrôle d'accès discrétionnaire (DAC)

II.4.1.1. Définition

Le Contrôle d'accès discrétionnaire (DAC pour Discretionary Access Control) est un genre de contrôle d'accès, développé par le TCSEC (Trusted Computer System Evaluation Criteria), qui définit le contrôle d'accès discrétionnaire comme : "un moyen de restriction d'accès aux objets basé sur l'identité des sujets et/ou groupes auquel ils appartiennent. Les contrôles sont discrétionnaires dans le sens où le sujet est capable de transférer les permissions d'accès à d'autres sujets".

Le modèle de contrôle d'accès discrétionnaire représente les politiques de contrôle d'accès sous forme d'un triplé « utilisateur, objet, action » qui exprime que l'utilisateur peut effectuer une certaine opération identifiée par l'action (ex., lire) sur l'objet spécifié. Le triplet est appelé une règle d'autorisation [14,15].

L'approche DAC permet au créateur d'un objet, ou toute autre personne qui est autorisée àle contrôler, de prendre des décisions de contrôle d'accès. Ces droits changent de façon dynamique à la discrétion du propriétaire d'un objet.

II.4.1.2. Modèle de Lampson

La notion des droits d'accès spécifiés par une matrice de contrôle d'accès a été introduite par Lampson dès 1971. Ce modèle peut être représenté par un triplet (S, O, Mso) où S désigne les sujets, O les objets et Mso la matrice de contrôle d'accès qui associe à chaque couple (sujet s, objet o) un ensemble de droits d'accès [16].

II.4.1.3. Modèle de Harrison RuzzoUllman

Le modèle de Harrison RuzzoUllman représente une amélioration du modèle de Lampson. Ce modèle de control d'accès fournit des commandes pour attribuer les droits d'accès (read, write, own, etc.), ainsi que pour créer et supprimer les sujets et les objets. Ce modèle est spécifié utilisant une matrice Mso avec s ϵ S, o ϵ O et MsoϵA [17].

II.4.1.4. La théorie et La pratique

Cependant, la signification de la limite n'est pas dans la pratique aussi définie que la définition donnée dans la norme TCSEC. Par exemple, la limite est utilisée généralement dans les contextes qui supposent que, sous DAC, chaque objet a probablement un propriétaire qui commande les permissions d'accéder à l'objet, parce que beaucoup de systèmes appliquent le DAC en utilisant le concept d'un propriétaire. Mais la définition de TCSEC n'indique rien au sujet des propriétaires, techniquement un système de contrôle d'accès ne doit pas avoir un concept de propriétaire pour concilier la définition du TCSEC et du DAC.

Autre exemple, des possibilités sont parfois décrites en tant que commandes discrétionnaires parce qu'elles permettent à des sujets de transférer leur accès à d'autres sujets, quoique la sécurité ne soit fondamentalement pas au sujet de l'accès basé sur l'identité des sujets. Des possibilités permettent généralement à des permissions d'être passées à n'importe quel autre sujet.

Pour passer ces permissions, il doit d'abord avoir accès au sujet de réception, et les sujets n'ont pas généralement accès à tous les sujets dans le système.

II.4.2. Modèle de contrôle d'accès obligatoire (MAC)

Afin de remédier au problème de fuites d'information des modèles de contrôle d'accès discrétionnaires, les modèles obligatoires (Mandatory Access Control) fixent des règles incontournables destinées à forcer le respect des exigences de contrôle d'accès. Ainsi, le modèle multi-niveaux affecte aux sujets et aux objets des niveaux de sécurité non modifiables par les utilisateurs et, par conséquent, limite leurs pouvoirs dans la gestion des accès à leurs données.

Le premier modèle, appelé modèle de Bell et La Pudula, a été développé pour le département de la défense américaine et vise, plus particulièrement, assurer la confidentialité. Le deuxième modèle, appelé modèle de Biba, s'est intéressé à l'intégrité. D'autres modèles

obligatoires, moins formalisés, ont été développés pour les systèmes commerciaux (Modèle de Clark et Wilson) et pour les institutions financières britanniques (Modèle de muraille de Chine) [14].

II.4.2.1. Le modèle Bell LaPadula

Le modèle de Bell LaPadula traite avec des droits d'accès comme lire ou écrire. Il étiquette aussi des sujets et des objets selon un classement de sécurité prédéfini. Ce modèle est légèrement plus flexible dans des organisations. L'accès en lecture est accordé aux sujets ayant une cote de sécurité plus élevés que les objets accessibles, et les sujets ayant des niveaux de sécurité inférieurs ils ont accès en écriture aux objets de classifications de sécurité plus élevés. La classification d'un sujet s est représentée par fs(s), et la classification d'unobjet o par fo(o). Ce modèle contient deux types d'identifiants de sécurité :

- Un identifiant hiérarchique : Cet identifiant représente le niveau de classification pour les objets (c'est-à-dire son niveau de sensibilité) ; et le niveau d'habilitation pour les sujets, qui sont typiquement :unclassified, confidential, secret, top secret. Ces niveaux de classification sont classés sous forme hiérarchique [18,19].

Top secret >secret>confidential>unclassified

- Des identifiants de catégories : Ces différentes catégories d'informations correspondent aux différentes organisations manipulant les données, par exemple militaire, privé, public. Ces identifiants sont indépendants de la hiérarchie de confidentialité.

Militaire >privé>public

- **La propriété simple**

La propriété simple du modèle Bell LaPadula peut être décrite tout simplement par la phrase « ne pas lire en haut ». En effet, cette propriété interdit à un sujet d'accéder en lecture à un objet qui a une classification plus élevée que l'habilitation du même sujet. Cette règle assure la confidentialité de l'information. La propriété simple est satisfaite si, pour chaque triplet (s, o, read), read étant le droit de lecture, le niveau de l'habilitation du sujet s est supérieur ou égal à la classification de l'objet o. [20] Cette propriété peut être représentée par une expression logique comme suit :

$$read \in M_{so} \rightarrow fS(s) \geq fO(o)$$

Figure II.3. Propriété du modèle Bell LaPadula (Ne pas lire en haut)

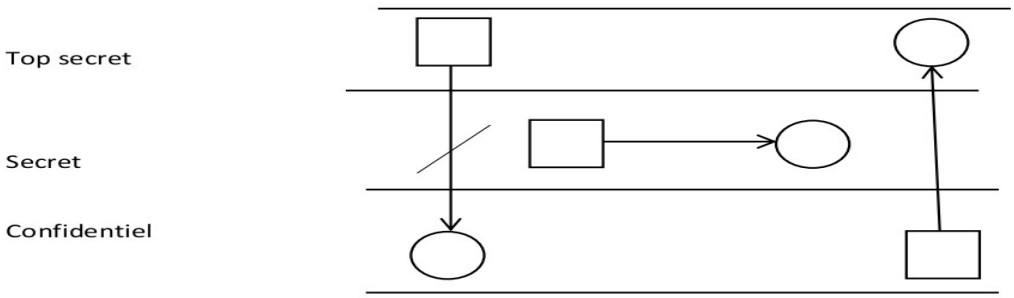

Figure II.4. Ne pas lire en haut Bell LaPadula

Les éléments qui ont été utilisés pour la représentation de la propriété simple du modèle Bell LaPadula :

- 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.

L'attribution de niveaux de sécurité aux objets et aux sujets comme indiqué dansla figure II.10 donne lieu aux règles suivantes :

- Un sujet qui a l'habilitation Confidentiel n'a pas le droit d'accéder en lecture à un objet qui a la classification Top Secret parce que le niveau de sécurité confidentiel est inférieur au niveau de sécurité Top Secret.
- Un sujet qui a l'habilitation Top Secret a le droit d'accéder en lecture à un objet qui a la classification Confidentiel parce que le niveau de sécurité Confidentiel est inférieur au niveau de sécurité Top Secret.
- un sujet qui a l'habilitation Secret a le droit d'accéder en lecture à un objet qui a la classification Secret parce que le même niveau de sécurité est attribué au sujet et à l'objet.
- **Propriété étoile**

La propriété étoile du modèle Bell LaPadula peut être décrite tout simplement par la phrase « ne pas écrire en bas ». En effet, cette propriété interdit à un sujet d'accéder en écriture à un objet qui a une classification moins élevée que son habilitation. Cette règle assure donc la prévention contre la divulgation d'informations. La propriété étoile est satisfaite si, pour chaque triplet (s, o, write), write étant le droit d'écriture, le niveau de l'habilitation du sujet s est inférieur ou égal à la classification de l'objet o. [20] Cette propriété peut être représentée par une expression logique comme suit :

$$write \in M_{so} \rightarrow fS(s) \geq fO(o)$$

Figure II.5. Propriété du modèle Bell LaPadula (Ne pas écrire en bas)

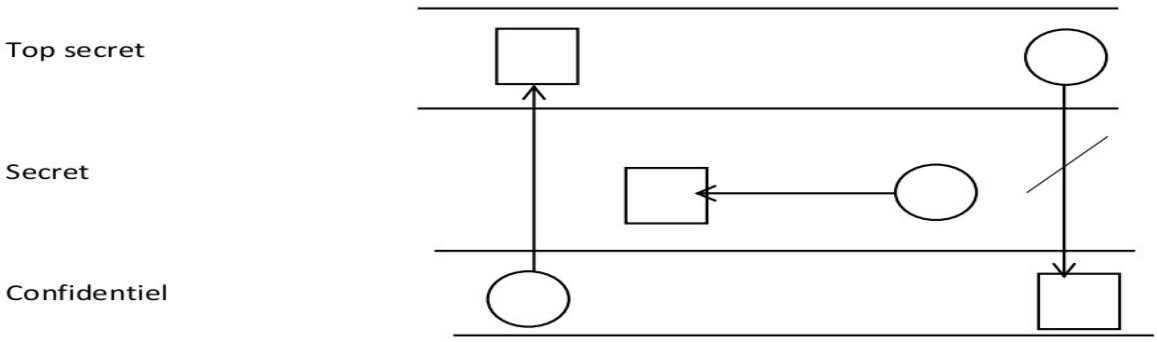

Figure II.6. Ne pas écrire en bas Bell laPadula

L'attribution de niveaux de sécurité aux objets et aux sujets comme indiqué dans la figure II.12 donne lieu aux règles suivantes :

- Un sujet qui a l'habilitation Confidentiel a le droit d'accéder en écriture à un objet qui a la classification Top Secret parce que le niveau de sécurité Confidentiel est inférieur au niveau de sécurité Top Secret.
- Un sujet qui a l'habilitation Top Secret n'a pas le droit d'accéder en écriture à un objet qui a la classification Confidentiel parce que le niveau de sécurité Confidentiel est inférieur au niveau de sécurité Top Secret.
- un sujet qui a l'habilitation Secret a le droit d'accéder en écriture à un objet qui a la classification Secret parce que le même niveau de sécurité est attribué au sujet et à l'objet.

II.4.2.2. Le Modèle Biba

Le modèle Biba vise à garantir l'intégrité. Il s'agit d'un modèle dual à BLP (Bell LaPadula). A chaque sujet et objet un classement de sécurité prédéfini leurs est associé. Ainsi, les règles relatives à la matrice de contrôle d'accès n'autorisent pas la modification du contenu d'un objet qu'aux sujets possédant un niveau d'intégrité suffisant. Tout comme le modèle Bell LaPadula ce modèle repose sur le respect de deux propriétés la propriété simple et la propriété étoile :

- La propriété simple

La propriété simple du modèle Biba peut être décrite tout simplement par la phrase « ne pas lire en bas ». En effet, cette propriété interdit à un sujet d'accéder en lecture à un objet qui a une classification moins élevée que l'habilitation du même sujet. La propriété simple est satisfaite si, pour chaque triplet (s, o, read), read étant le droit de lecture, le niveau de l'habilitation du sujet s est inférieur ou égal à la classification de l'objet o. [20] Cette propriété peut être représentée par une expression logique comme suit :

$$read \in M_{so} \rightarrow fS(s) \leq fO(o)$$

Figure II.7. Propriété du modèle Biba (Ne pas lire en bas)

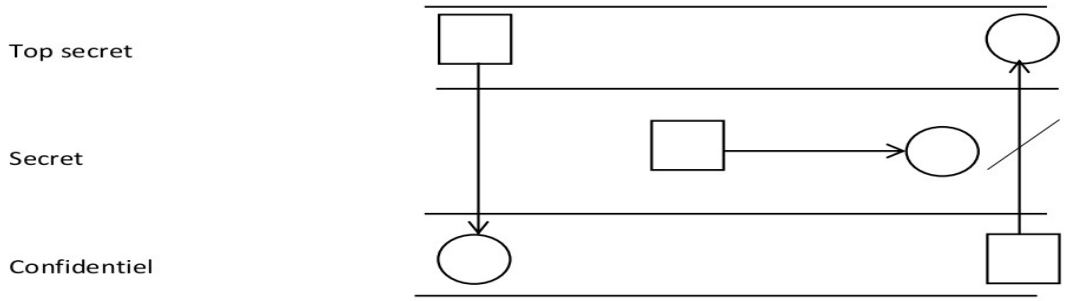

Figure II. 8. Ne pas lire en bas Biba

L'attribution de niveaux de sécurité aux objets et aux sujets comme indiqué dans la Figure II.14 donne lieu aux règles suivantes :

- Un sujet qui a l'habilitation Confidentiel a le droit d'accéder en lecture à un objet qui a la classification Top Secret parce que le niveau de sécurité Confidentiel est inférieur au niveau de sécurité Top Secret.
- Un sujet qui a l'habilitation Top Secret n'a pas le droit d'accéder en lecture à un objet qui a la classification Confidentiel parce que le niveau de sécurité Confidentiel est inférieur au niveau de sécurité Top Secret.
- Un sujet qui a l'habilitation Secret a le droit d'accéder en lecture à un objet qui a la classification Secret parce que le même niveau de sécurité est attribué au sujet et à l'objet.

- **La propriété étoile**

La propriété étoile du modèle Biba peut être décrite tout simplement par la phrase « ne pas écrire en haut ». En effet, cette propriété interdit à un sujet d'accéder en écriture à un objet qui a une classification plus élevée que son habilitation. La propriété étoile est satisfaite si, pour chaque triplet (s, o, write), write étant le droit d'écriture, l'habilitation du sujet s est supérieure ou égal à la classification de l'objet o. [20] Cette propriété peut être représentée par une expression logique comme suit :

$$write \in M_{so} \rightarrow fS(s) \geq fO(o)$$

Figure II.9. Propriété du modèle Biba (Ne pas écrire en haut)

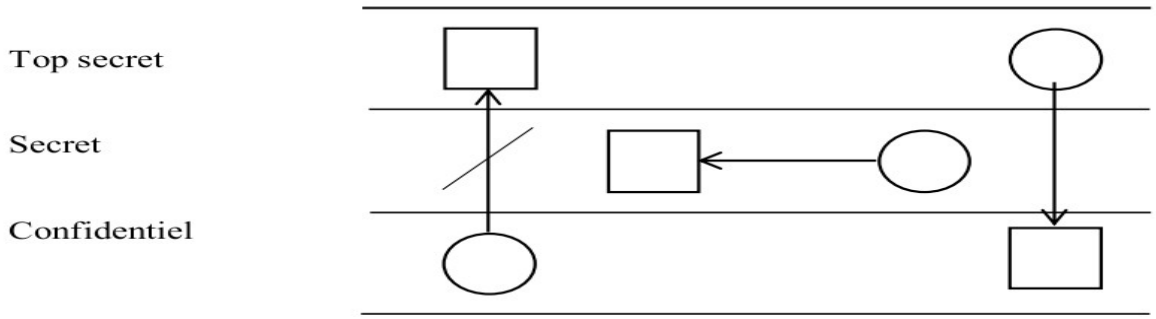

Figure II.10. Ne pas écrire en haut Biba

L'attribution de niveaux de sécurité aux objets et aux sujets comme indiqué dans la figure II.16 donne lieu aux règles suivantes : Un sujet qui a l'habilitation Confidentiel n'a pas le droit d'accéder en écriture à un objet qui a la classification Top Secret parce que le niveau de sécurité Confidentiel est inférieur au niveau de sécurité Top Secret.

- Un sujet qui a l'habilitation Top Secret a le droit d'accéder en écriture à un objet qui a la classification Confidentiel parce que le niveau de sécurité Confidentiel est inférieur au niveau de sécurité Top Secret.
- Un sujet qui a l'habilitation Secret a le droit d'accéder en écriture à un objet qui a la classification Secret parce que le même niveau de sécurité est attribué au sujet et à l'objet [20].

II.4.3. Contrôle d'accès basé sur les rôles RBAC

En raison de la rigidité des deux modèles DAC et MAC, il est difficile de leur intégrer la délégation. Pour cette raison, l'accent sera mis sur le contrôle d'accès basé sur le rôle (RBAC), ce qui peut être considéré comme une combinaison des deux MAC et approches du DAC.

Dans la plupart des organisations, une politique de sécurité doit être appliquée aux centaines, si pas des milliers, de salariés. Pour simplifier l'administration de sécurité, beaucoup d'organisations définissent des rôles auxquels des individus multiples peuvent être associés. La politique de sécurité de l'organisation définit alors comment les permissions doivent être associées à ces rôles. Sandhu et al présenté l'approche de RBAC qui est particulièrement efficace quand les changements sont faits à la politique de sécurité organisationnelle. Le modèle de RBAC doit seulement être fait aux attributions de rôle, qui sont significativement moins que des attributions individuelles [21]

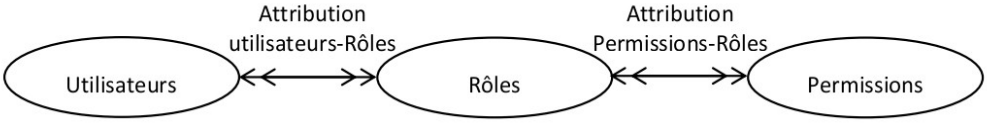

Figure II.11. Attribution des permissions en RBAC

Un rôle peut avoir plusieurs permissions et une permission peut être associée à plusieurs rôles. Un utilisateur peut jouer plusieurs rôles et un rôle peut être attribué à plusieurs utilisateurs.

II.5 Définition d'attaque

Une « attaque » est l'exploitation d'une faille d'un système informatique (système d'exploitation, logiciel ou bien même de l'utilisateur) à des fins non connues par l'exploitant des systèmes et généralement préjudiciables [22].

II.5.1. Certains modes d'attaque

Piratage

C'est l'accès non autorisé d'un système informatique pour obtenir des données ou des informations appartenant à quelqu'un d'autre. Le pirate exploite les faiblesses et vulnérabilités des systèmes. Il existe diverses voix par lesquelles un pirate informatique peut accéder à un système [23].

Les attaques au niveau application

Vue que les développeurs de logiciels sont sous pression afin de livrer des produits dans les délais, à cela s'ajoute l'utilisation des méthodes de génie logiciel. Les logiciels sont devenus très complexes et ont donné lieu à d'énormes quantités de caractéristiques et de fonctionnalités dans l'application, les besoins font parfois réduire les délais pour les essais et les testes et la mise en place d'outils de sécurité viennent un peu plus tard et sont livrés en tant que composants après implémentation.

- Les attaques des logiciels mal configurés.
- Les attaques des systèmes d'exploitation, injection de SQL, piratage de mot de passe.
- Hameçonnage, Ingénierie sociale. Sniffer /Mouchards. Les virus et les vers, les débordements de tampon.

II.6. Définition de vulnérabilité

- **Définition 1**

Une vulnérabilité est une faille de sécurité qui résulte de la conception du système informatique imprudente, mauvaise application, un mauvais entretien ou d'une exploitation inappropriée [24].

- **Définition 2**

La vulnérabilité du système est une lacune ou faiblesse dans la conception ou la mise en œuvre d'un système d'information (y compris les contrôles de sécurité et des procédures de sécurité associées au système) qui pourraient être intentionnellement ou non exploitée pour affecter une organisation ses activités ou actifs par le biais d'une perte de confidentialité, d'intégrité ou de disponibilité [25].

Certaines vulnérabilités du système/réseau incluent les éléments suivants :

- Mauvaise configuration du système.
- Un accès facile à l'information.
- L'absence de politiques de sécurité appropriées.
- Manque d'efficacité de la sécurité physique.

Des vulnérabilités peuvent être évité si certain mesures sont mises en place, ces mesures sont des mécanismes non fonctionnels qui sont intégrés dans le processus de développement de l'application, fonctionnel comme l'utilisation des antivirus et firewall ainsi que des bonnes pratiques des utilisateurs.

II.7. Quelques consignes de sécurité

1. Pour éviter l'injection SQL.
   - Réduire les privilèges de connexions à la base de données.
   - Sauvegarder le système compte.
2. Pour empêcher un craquage de mots de passe.
   - Choisissez les mots de passe de moins de huit caractères.
   - Les mots de passe doivent avoir une combinaison de lettres majuscules et minuscules, de chiffres, de caractères etc. Ceci le rend difficile à craquer.
3. Pour éviter une surcharge de la mémoire tampon :
   - Mettre en œuvre manuels d'audit de codes.
   - Désactiver l'exécution de la pile.
   - Adopter les techniques efficaces et robustes [26].

II.8. Conclusion

Dans ce chapitre on a traité une partie bien détaillé concernant les modèles de contrôles 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 donnée, ainsi Nous avons précédemment présenté les types d'attaques et de vulnérabilité. Enfin, nous avons présenté notre but du projet et les argumentations sur le choix des diagrammes et du modèle de contrôle d'accès.

Dans ce qui va suivre nous allons établir la partie modélisation, analyse et conception de notre application.

III.1. Introduction

Dans ce chapitre nous allons présenter la partie modélisation de notre application, pour cela nous allons passer par trois parties essentielles, dans la première partie nous allons définir l'application et argumenter les choix théoriques, dans la deuxième partie nous proposons les algorithmes avec les résultats de leurs déroulements (deux exemples seront déroulés), et enfin nous allons présenter l'analyse et la conception de l'application.

III.2. Définition de l'application

Notre application permet de détecter les failles dans un système modélisé en UML (Diagramme de composant, diagramme de déploiement étendu).

Pour cela nous allons offrir à l'utilisateur une interface adéquate pour introduire (gérer) les éléments du modèle d'application (diagramme de composant et diagramme de déploiement étendu) et la politique de contrôle d'accès basée sur le DAC; en se basant sur ces données, le système détecte les failles en utilisant la théorie des graphes.

III.2.1. But de l'application

Cette application permet de détecter les failles d'un système dans la phase modélisation (avant la phase d'implémentation) d'où un gain de temporel et financier, elle a comme but :

- Détecter le problème lié au contrôle d'accès.
- Détecter le problème du goulot d'étranglement à cause de la vitesse des nœuds.
- Détecter le problème de congestion causé par le débit (internet) de la bande passante.
- Détecter les nœuds vulnérables (deux ou plusieurs réseaux sont accordés par un seul nœud).

III.2.2. Pourquoi avons-nous choisi les deux diagrammes composant et déploiement ?

Les diagrammes de composant et de déploiement portent une vue structurelle sur le système, notre choix s'est porté sur ces deux diagrammes pour les raisons suivantes:

- Les deux diagrammes permettent d'illustrer l'aspect physique du système.

- Les deux diagrammes sont liés par le faite que le diagramme de déploiement décrit la disposition des composants dans les nœuds.
- Les acteurs sont modélisés sur diagramme de déploiement ce qui permet de répondre a la question importante : de ce qui fait? (acteur) quoi ? (interaction avec les composants) et où dans le système (les nœuds)?

III.2.3. Pourquoi avons-nous choisi le modèle DAC ?

Dans le diagramme de déploiement étendu les acteurs sont liés à des nœuds et sur chaque nœud des composants sont repartis, ce qui permet de déduire la liaison entre composant et acteur.

Si on considère les acteurs (modèle application) comme des sujets actifs qui interagissent avec le système et les composants (modèle application) comme des fichiers (sources, données…) alors la politique de contrôle d'accès DAC s'impose.

Pour préciser la politique de contrôle d'accès l'utilisateur de notre application doit avoir en plus de son modèle d'application une politique de contrôle d'accès définie sous la forme de qui (acteur) exécute (ou n'exécute pas) quoi (composant) dans le système.

III.2.4. Pourquoi avons-nous choisi la théorie des graphes ?

Afin de détecter les failles dans un système nous nous somme appuyés sur la théorie des graphes comme moyen formel pour la recherche et la localisation des vulnérabilités et cela à cause de :

- La nature des données des deux diagrammes du modèle d'application (les éléments des deux diagrammes sont représentés par des sommets et les relations entre ces éléments par des arcs)
- La puissance et la clarté des algorithmes
- La facilité de représenter des données (sous forme matricielles)
- Des résultats fiables.

III.3. Vue statique de haut niveau de l'application

Notre projet est divisé structurellement en trois packages comme le montre le schéma III.1 ci-dessous.

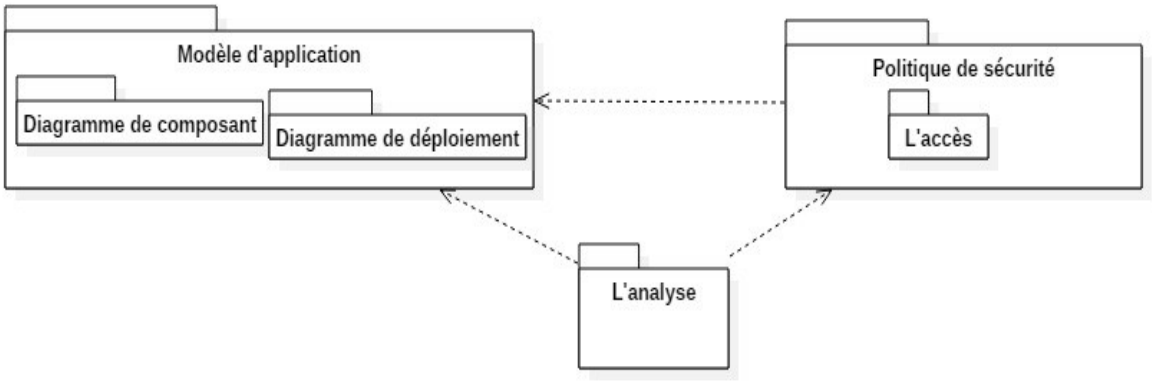

Figure III.1. Diagramme de package de haut niveau de l'application DAC.

a) **Modèle d'application** : ce package regroupe les propriétés du diagramme de composant ainsi que le diagramme de déploiement.
b) **Politique de sécurité** : ce package regroupe les données qui permettent d'introduire les éléments de la politique de sécurité ces données dépendent du modèle d'application.
c) **Analyse** : ce package regroupe essentiellement les algorithmes qui permettent d'analyser et détecter les failles, les paramètres d'entrée de ces algorithmes dépendent du modèle d'application et de la politique de sécurité.

III.4. Vue dynamique de haut niveau de l'application

Afin de bien comprendre notre application nous proposons une vue dynamique de haut niveau, cette vue est illustrée par le diagramme d'activité figure III.2, ce diagramme montre l'enchaînement des actions et décisions au sein d'une activité gérée par l'acteur.

- **Diagramme d'activité de haut niveau de l'application**

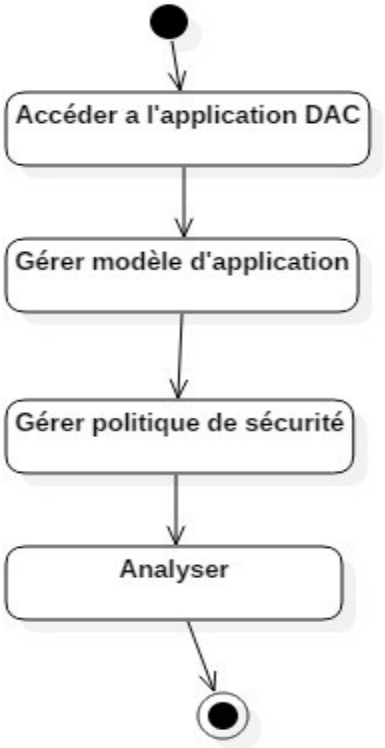

Figure III.2. Diagramme d'activité de haut niveau de l'application.

Dans ce qui va suivre nous allons détailler chaque activité par un diagramme d'activité approprié et des algorithmes.

III.4.1. Diagramme de haut niveau gérer modèle d'application de l'application

Ce diagramme figure III.3 montre une succession d'étapes pour gérer (ajouter, modifier, supprimer les éléments) le modèle d'application cette gestion impose à l'utilisateur de commencer par la gestion du diagramme de composant suivi par la gestion du diagramme de déploiement.

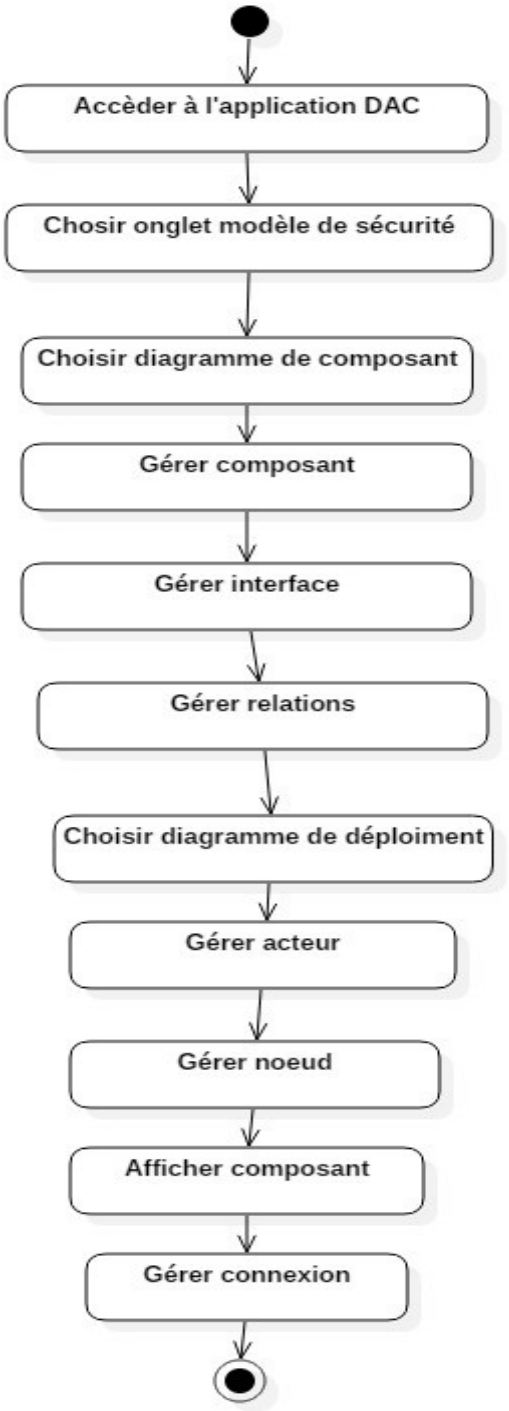

Figure III.3. Diagramme d'activité de haut niveau gestion du modèle d'application.

III.4.2. Diagramme d'activité de la politique de sécurité de haut niveau

Ce diagramme figure III.4 montre une succession d'étapes pour gérer (ajouter, modifier, supprimer les éléments) la politique de sécurité. Cette gestion commence par le choix d'une des deux variantes de la politique de contrôle d'accès : les autorisations (ce qui est permis pour un acteur) ou les interdits (ce qui est interdit pour un acteur), ensuite l'utilisateur introduit le composant et l'acteur, et enfin valide ce choix.

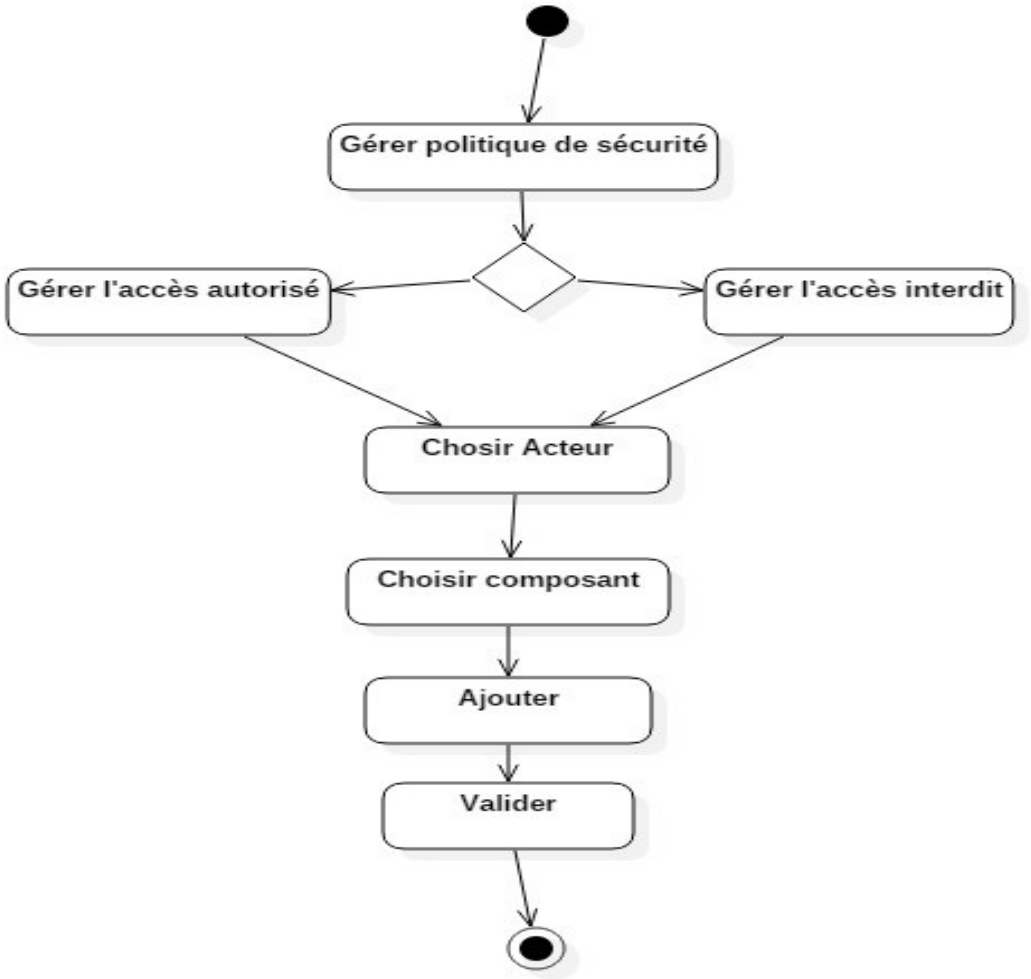

Figure III.4. Diagramme d'activité de haut niveau gérer politique de sécurité.

III.4.3. Diagramme d'activité de haut niveau d'analyse

Ce diagramme figure III.5 montre une succession d'étapes pour analyser, détecter et afficher les problèmes de sécurité. Ces étapes montrent une séquence d'algorithme (fonction de haut niveau) qui permettent de déceler les failles du système.

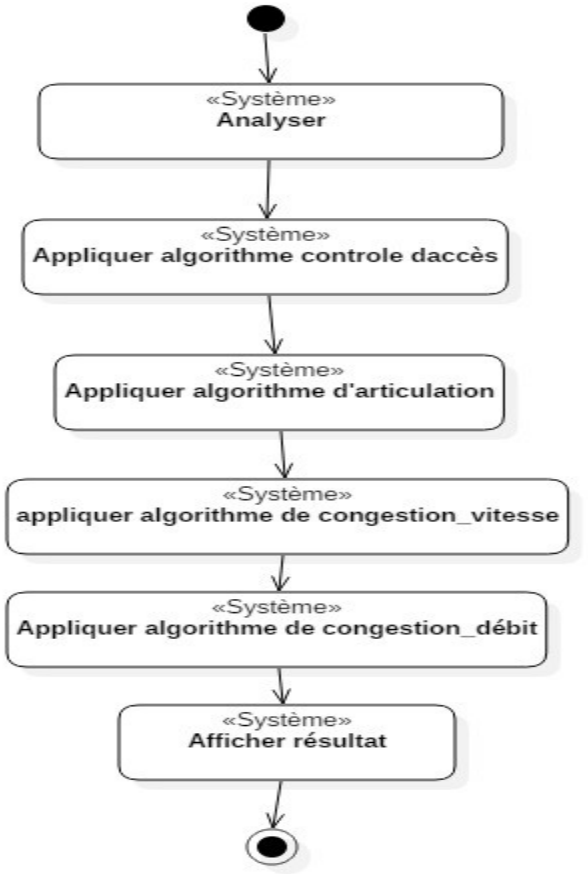

Figure III.5. Diagramme d'activité de haut niveau analyse.

Ces algorithmes sont :

- **Algorithme contrôle d'accès** : cet algorithme permet de détecter la présence d'une violation de politique de contrôle d'accès, il se base sur la matrice composant (calculée à partir du diagramme composant) et la matrice acteur composant (calculée à partir du diagramme de déploiement), et la matrice des interdits (calculée à partir de la politique de sécurité)
- **Algorithme d'articulation** : cet algorithme permet de détecter les nœuds vulnérables dans le diagramme de déploiement, ces nœuds représentent une

faille majeure pour l'application car une attaque réussie pour ce nœud n'affecte pas la machine en question mais divise le réseau en deux sous réseaux.

- **Algorithme de congestion- vitesse :** cet algorithme permet de détecter le problème le goulot d'étranglement à cause de la vitesse des nœuds
- **Algorithme de congestion- débit :** il permet de détecter le problème Détecter le problème de congestion à cause de la bande passante.

Dans ce qui va suivre nous allons exposer nos algorithmes, deux exemples illustratifs seront exposés pour le déroulement.

Exemple d'un diagramme de composant et de déploiement étendu :

- **Diagramme de composant**

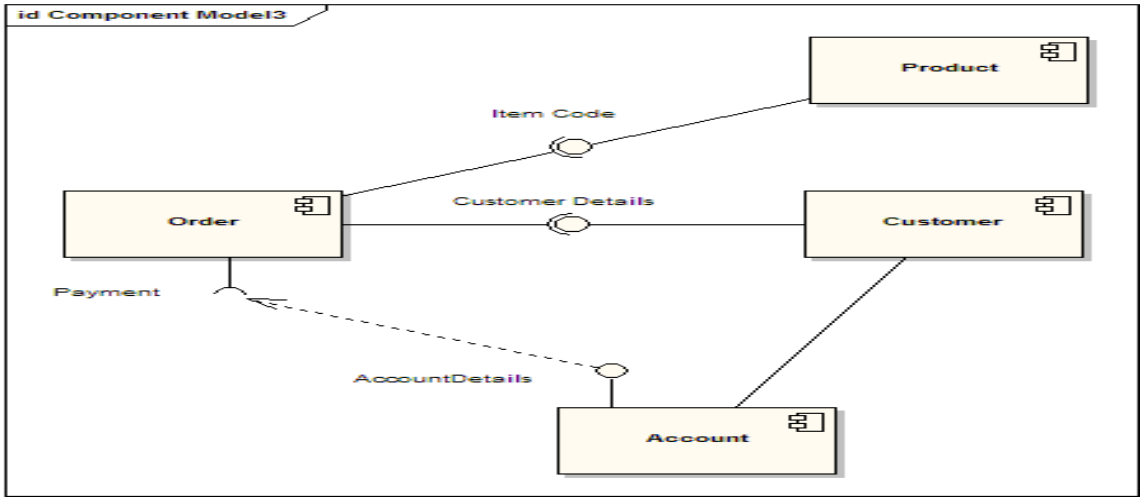

Figure III.6 Diagramme de composant.

Cet exemple figure III.6 montre un diagramme de composant contenant 4 composants Order, Product, Customer, Account, ces composants sont en relation de dépendance ainsi Order dépend de Product, Custumer et Account et Account dépend du custumers.

- **Diagramme de déploiement**

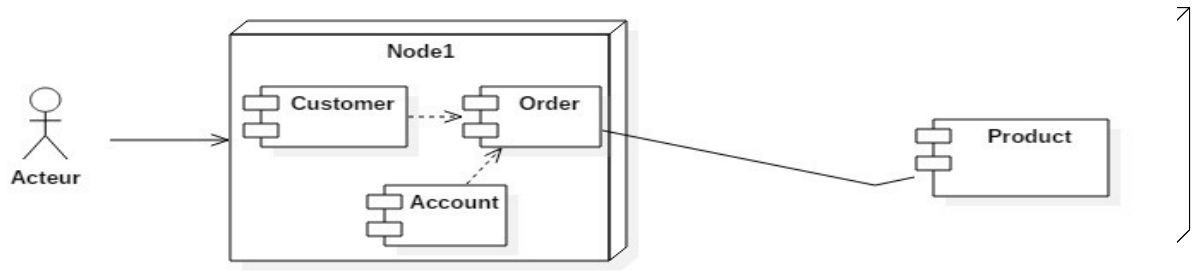

Figure III.7. Diagramme de déploiement.

Ce diagramme montre un acteur qui interagit avec un nœud Node1 qui dispose des composants Custumer, Order ,Account, et Node 2 qui inclus le composant Pruduct.

III.5. Algorithmes Transformant les diagrammes en matrices

- **Algorithme de dépendance Composant_Composant :** le tableau III.1 montre l'algorithme qui permet de déduire la matrice de dépendance entre les composants (Marice_CC) a partir du diagramme de composant.

Tableau III.1. Algorithme de dépendance Composant_Composant.

```
Fonction matrice C_C(Nb_composant)
Entier i,j;
Début
Pour i= 0  à Nb_composant Faire
      pour j= 0 à Nb_composant Faire
                        Si (i==j) alors
                                    Matrice CC[i][j]=0;
                        Si non
                                Si (FindII(i,j)=1) alors
                                                    Matrice _CC[i][j]=1;
                                Si non
                                                   Matrice_CC[i][j]=0;
                                Fin si
                        Fin si
          Fin pour
Fin pour
Fin
```

Cette fonction ci dessus fait appel à la fonction FindII :

- **Algorithme de dépendance Interface_Interface**.

Tableau III.2. Algorithme de dépendance Interface_Interface.

```
Fonction FindII(composant1,composant2)   //a comp1,b comp2
Entier i,j ;
Début
Pour j = 0 à Nb_Interface Faire
                Si(Composant-Interface[composant1][j]=1)  alors //que a est relié à
Interface j
                  Pour i de 0 à Nb_Interface faire
                                Si (i=j) alors
                                                Continue
                                Fin si
                                Si(Composant-Interface[composant2][i]=1) alors
                              Si(Interface-Interface[i][j]=1 ou Interface-Interface[j][i]=1 )
alors
                                                                                Retourner (1) ;
                          Fin si
                   Fin si
                   Fin pour
             Fin si
Fin pour                Fin
```

Après avoir appliqué l'algorithme Fonction matrice C_C(Nb_composant) sur l'exemple, on déduit la matrice composant composant. Le tableau III.3 montre le résultat du déroulement.

Tableau III.3 Matrice CC de dépendance (composant composant)

Dep C/C	Order	Product	Customer	Account
Order	0	1	1	1
Product	1	0	0	0
Customer	0	0	0	0
Account	0	0	0	0

- **Algorithme de dépendance Acteur_Composant :** le tableau III.4 montre l'algorithme qui permet de déduire la matrice acteur composant à partir du diagramme de déploiement.

Tableau III.4 Algorithme de dépendance Acteur_Composant

```
Fonction A_c (Nb-acteur,Nb-Composant)
Entier k,l,i;
Début
Pour i= 0 à Nb_acteur Faire
        Pour j= 0 à Nb_composant Faire
                Si (ChercherNC(k,l)==1 ) alors
                                                Matrice_AC[k][l]=1;
                Si non Matrice_AC[k][l]=0
                Fin si
         Fin pour
Fin pour
Fin
```

Après avoir appliqué l'algorithme Fonction A_c (Nb-acteur,Nb-Composant) sur l'exemple, on déduit la matrice acteur composant. Le tableau III.5 montre le résultat du déroulement.

Tableau III.5 Matrice Acteur composant.

Dep Acteur /Cj	**Customer**	**Account**	**Order**	**Product**
Acteur	**1**	**1**	**1**	**0**

- **L'algorithme utilisé pour la politique de contrôle d'accès :** notre application offre la possibilité de saisir la politique de sécurité selon deux variantes : ce qui est interdit et ce qui est autorisé, ces deux variantes nous mènent vers deux matrices différentes : la matrice des autorisés ou matrice des interdits, toute fois pour des raisons d'optimisation nous allons transformer la matrice des autorisés en matrice des interdits l'algorithme suivant est utilisé dans le cas ou l'utilisateur choisit la variante de politique de contrôle d'accès autorisé alors la matrice des interdits est déduite tableau III.6.

Tableau III.6. Algorithme qui déduit la matrice des interdits

```
fonction dual_autorisé (Matrice_autorisé[][],Nb_Composant)
Entier i,j
Début
Pour i = 0 à Nb_acteur Faire
        Pour i = 0 à Nb_Composant Faire
                Si (Matrice_autorisé[i][j]==1) alors
                                        Matrice_interdit [i][j]=0;
                Si non
                        Matrice_interdit [i][j]=1;
                Fin si
          Fin pour
Fin pour
Fin
```

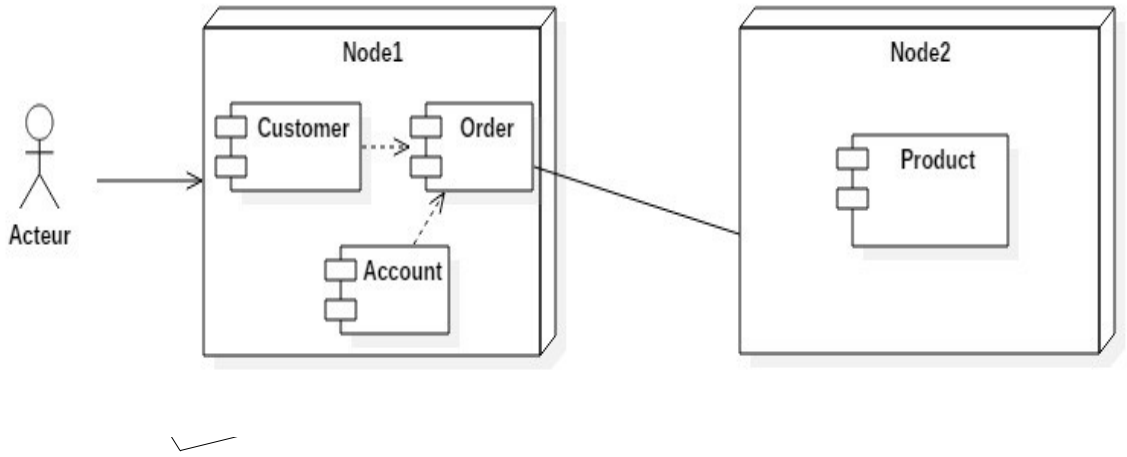

Figure III.8. Contrôle d'accès interdit

Le tableau 8 montre une politique de contrôle d'accès avec la contrainte que l' acteur ne doit pas accéder au composant product comme le montre la figure III.8.

Tableau III.7. Matrice Acteur composant

Accés interdit	Customer	Account	Order	Product
Acteur	0	0	0	1

III.6. Les algorithmes de détection des failles

- **Algorithme détecte faille contrôle d'accès:** le tableau III. 7 montre l'algorithme qui permet de détecter les faille lié au accès non autorisé.

Tableau III.8. Algorithme détecte faille contrôle d'accès (Interdit)

```
Fonction Con_Acc (Matrice_interdit [][], Matrice-Acteur-composant[][])
Entier Nb_Acteur,i,j;
Debut
Pour i = 0 à Nb_Acteur faire
  Pour j =0 à Nb_Composant faire
          Si(Matrice_interdit [i][j]=0 et Matrice_AC [i][j]=1)  alors
                              Si(type-Composant[j]="Application") alors
                                                  Afficher (''Détection de faille '')
                              Finsi
              Finsi
    Finpour
Finpour
Fin
```

Cet algorithme détecte les failles en Comparant la matrice Acteur-Composant avec la matrice Interdit.

Dans l'exemple précédent une faille est détectée entre acteur et le composant product. Cette détection est dû au faite l'acteur peut accéder au composant Product à travers la dépendance entre les composants (relation indirecte), alors que la politique de sécurité lui interdit.

- **Exemple pour le déroulement des algorithmes**

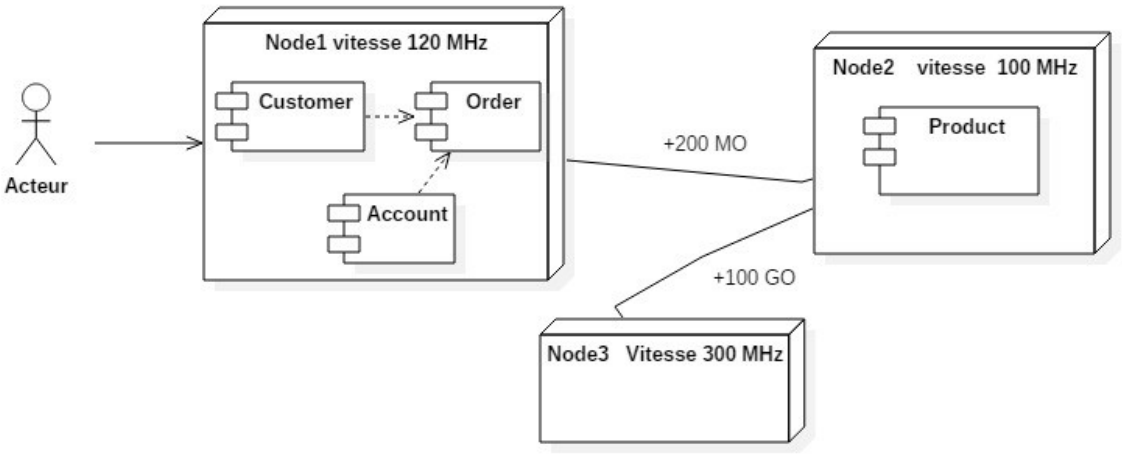

Figure III.9 . Diagramme de déploiement étendu (problème de conestion_vitesse).

- **Algorithme d'articulation :** le tableau III.9 montre l'algorithme qui permet de repérer les nœuds d'articulation, en comptant le nombre de ces nœuds qui lui sont reliés.

Tableau III.9 Algorithme d'articulation.

```
Fonction articulation (matrice_NN[] [], Nb_noeud)
Entier i,j, Nb_conexion;
Debut
pour i = 0 à Nb_Noeud faire
  pour j =0 à Nb_Noeud faire
                Si ((matrice_NN [i][j]=1)Alors
                                Nb_conexion = Nb_conexion +1
              Finsi
  FinPour
  Si (Nb_conexion = 2) alors
           Afficher ("Le Noeud i est un point d'articulation")
   Finsi
Finpour   Fin
```

En appliquant cet algorithme nous aurons comme résultat de vulnérabilité:

Attention le node2 est un point d'articulation.

- **Algorithme de congestion_vitesse :** le tableau III.10 montre l'algorithme qui détecte les nœuds qui permettent de créer le problème de congestion lié à leur vitesse d'exécution (exemple: fréquence processeur).

Tableau III.10 Algorithme de congestion_vitesse.

```
fonction Congestion_Vitesse(matrice NN,Nb_Noeud, seuil_vitesse)
Entier i,j;
Début
Pour i = 0 à Nb_Noeud faire
          Pour j = 0 à Nb_Noeud faire
                    Si(matrice NN[i][j]==1) alors
                                        Si  ((|Nvitesse[i]-Nvitesse[j]|)
>seuil_vitesse)alors
                    afficher ("une faille a été détecté entre le noeud i et
j")
                              Fin si
                     Fin si
          Fin pour
Fin pour
```

Avec un seuil de vitesse =100MHz, nous aurons comme résultat un problème de congestion dans le Node2.

- **Algorithme de congestion_vitesse :** le tableau III.11 montre l'algorithme qui permet de détecter les nœuds qui permettent de créer le problème de congestion lié a leur connexion (exemple: bande passante des liens entre les nœuds).

Tableau III.11 Algorithme de congestion-débit.

```
Fonction congestion_débit(matrice NN[] [],Nb_Noeud, SeuilDebit)
Entier i,j,k;
Debut
Pour i = 0 à Nb_Noeud Faire
  Pour j = 0  à Nb_Noeud Faire
        Si((Debit-Noeud[i][j]>0) alors
              Pour k = 0  à Nb_Noeud faire
                    Si((Debit-Noeud[i][k]>0) alors
             Si((|Debit-Noeud[i][j]-Debit-Noeud[i][k]|) >SeuilDebit)  alors
                          Afficher("Détection de faille pour le nœud : i")
               Finsi
            Finpour
Finpour
Fin
```

En appliquant l'algorithme de Congestion_débit(), avec un seuil SeuilDebit = 50 sur l'exemple précédent nous aurons comme résultat ‘’une faille à été détectée au niveau de node2 ‘’

III.7. Cahier des charges fonctionnel

1. L'utilisateur introduit le modèle de son application :
   a) L'utilisateur saisit les différentes propriétés des diagrammes de composant.
   b) L'utilisateur saisit les différentes propriétés des diagrammes de déploiement.
2. L'utilisateur introduit la politique de sécurité de son application :
   a) L'utilisateur introduit la politique de contrôle d'accès selon le modèle DAC.
3. Le système analyse la cohérence entre le modèle de l'application et la politique de contrôle d'accès.

III.8. Analyse et conception

Le succès du développement du logiciel dépend évidemment de la bonne utilisation d'une méthode mais il dépend surtout de la façon dont on utilise cette méthode à l'intérieur du cycle de développement du logiciel.

Le processus que nous vous proposons de suivre pour notre projet est le RUP(Rational Unified Process).

III.8.1. RUP

RUP (Rational Unified Process) est un processus basé sur une approche disciplinée afin de bien maitriser l'assignation des taches et la responsabilisation des différents acteurs participant au cycle de développement du logiciel [27,28].

Selon notre projet nous avons eu besoin de modéliser seulement six diagrammes parmi les 14 proposés dans le langage UML.

Chapitre 3 : Analyse et conception

Notre modélisation passe par deux parties importantes analyse et conception

1. **L'analyse des besoins** : qui se divise en quatre étapes :
   a) La collecte des besoins.
   b) La spécification des cas d'utilisation.
   c) Description textuelle de chaque cas d'utilisation.
   d) Diagramme de classe d'analyse.
2. **La conception**: elle se divise en quatre étapes :
   a) La spécification de diagramme de classe de conception.
   b) La spécification de diagramme de séquence de conception.
   c) La spécification de diagramme de composant
   d) La spécification de diagramme de déploiement
3. **L'implémentation** : Cette phase sera décrite dans le chapitre suivant.

III.8.2. Analyse

III.8.2.1. Diagramme de cas d'utilisation d'application

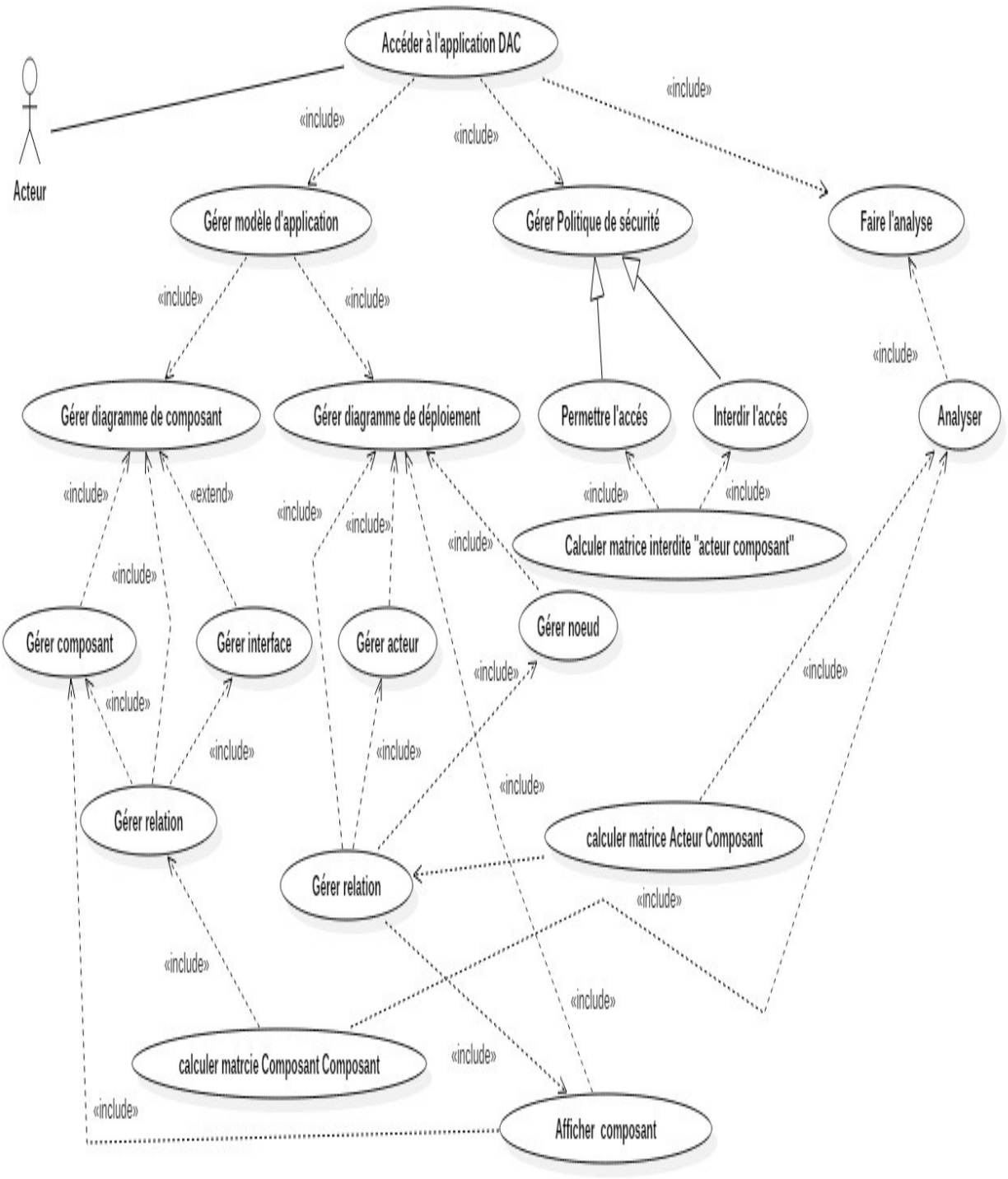

Figure III.10. Diagramme de cas d'utilisation de notre application.

III.8.2.1.1. Description textuelle de diagramme de cas d'utilisation de l'application

- **Description textuelle : gérer diagramme de composant**

Titre : Gérer composant.
Résumé : ce cas d'utilisation permet d'ajouter, modifier et supprimer des composants.
Acteur : utilisateur. **Responsable**: Baghdadi Sihem
Pré condition : Application déjà sur papier.
Scénario nominal 1 : Gérer composant.

Scénario 1 : Ajouter un composant.

1-Accéder à l'application DAC.
2-Afficher l'onglet du diagramme de composant.
3-Choisir onglet composant.
4-choisir onglet Propriété.
5-saisir le nom du composant.
6-saisir le type du composant.
7-saisir le stéréotype.
8-cliquer sur ajouter.

Scénario 2 : Modifier un composant.
Pré condition : composant existe déjà.
Modifier un composant :
1- Accéder à l'onglet composant.
2-Afficher l'onglet liste de composant.
3-Sélectionner un composant.
5-cliquer sur modifier.
6-Afficher l'onglet propriétés.
7-Saisir à nouveau les nouvelles propriétés.
8- Cliquer sur valider.
Post condition : composant modifié.

Scénario 3 : Supprimer un composant.
Pré condition : composant existe déjà.
Supprimer un composant :
1-Accéder à l'onglet composant.
2-Afficher l'onglet liste de composant..
3-Sélectionner un composant.
5-cliquer sur supprimer.
Post condition : composant supprimé.

Scénario d'erreur :
Scénario 1 : champ vide :
Etape 1 : Affichage d'une boite de dialogue ''Vous devez remplir tous les champs svp''.
Etape 2 : Retournez au scénario 1'étape 5.

Scénario 1 : Composant dupliqué :
Etape 1 : Affichage d'une boite de dialogue ''ce composant existe déjà''.
Etape 2 : Retournez au scénario 1'étape 5.
Scénario Alternatif : Aucun.

- **Description textuelle : gérer interface**

Titre : Gérer interface.
Résumé : ce cas d'utilisation permet d'ajouter, modifier et supprimer des interfaces.
Acteur : utilisateur. **Responsable**:BaghdadiSihem
Pré condition : Application déjà sur papier.
Scénario nominal 2 : Gérer interface.
Scénario 1 : Ajouter une interface.
1-Accéder à l'application DAC.
2-Afficher l'onglet du diagramme de composant.
3-Choisir onglet interface.
4-choisir onglet Propriété.
5- saisir le nom.
6-saisir le type.
7-cliquer sur ajouter.

Scénario 2 : Modifier une interface.
Pré condition : interface existe déjà dans la liste.
Modifier une interface:
1- Accéder à l'onglet interface.
2-Afficher l'onglet liste des interfaces.
3-Sélectionner une interface.
5-cliquer sur modifier.
6-Afficher l'onglet propriétés.
7-Saisir à nouveau les nouvelles propriétés.
8- Cliquer sur valider.
Post condition : interface modifiée.

Scénario 3 : Supprimer une interface.
Pré condition : composant existe déjà dans la liste.
Supprimer une interface :
1-Accéder à l'onglet interface.
2-Afficher l'onglet liste des interfaces.
3-Sélectionner une interface.
5-cliquer sur supprimer.
Post condition : interface supprimé.
Scénario d'erreur :
Scénario 1 : champ vide :
Etape 1 : Affichage d'une boite de dialogue ''Vous devez remplir tous les champs svp''.

Etape 2 : Retournez au scénario 1'étape 5.
Scénario 2 : Composant dupliqué :
Etape 1 : Affichage d'une boite de dialogue ''cette interface existe déjà''.
Etape 2 : Retournez au scénario 1'étape 5.
Scénario Alternatif : Aucun.

- **Description textuelle : gérer relation**

Titre : Gérer relation dans un diagramme de composant.
Résumé : ce cas d'utilisation permet d'ajouter, modifier et supprimer les connexions entre les différents intervenants du diagramme de composant.
Acteur : utilisateur. **Responsable**:BaghdadiSihem
Pré condition : liste de composant et de déploiement déjà remplie.
Scénario nominal: Gérer relation.
Scénario 1 : Ajouter une relation.
1-Accéder à l'application DAC.
2-Afficher l'onglet du diagramme de composant.
3-Choisir onglet relation.
4-Choisir onglet la création.
5-saisir le nom.
6-saisir le type.
7-choisir interface ou composant de départ.
8-choisir interface ou composant d'arrivé.
9-cliquer sur ajouter.
10-Générer les matrices.
Scénario alternatif :
Le système n'acceptera pas de laisser un champ vide il affiche ''svp vous devez remplir tous les champs''.
Le système n'acceptera pas un nom écrit deux fois.
Post condition : relation ajoutée dans la liste des relations, affichage de relations dans la liste de dépendance et la génération des matrices.

Scénario 2 : Modifier une relation.
Pré condition : relation existe déjà dans la liste des relations.
Modifier une relation de la liste des relations:
1- Accéder à l'onglet liste des relations.
2-Sélectionner une relation.
5-cliquer sur modifier.
6-Afficher l'onglet relation.
7-Saisir à nouveau les nouvelles propriétés sur l'onglet ''la création''.
8- Cliquer sur valider.
Post condition : relation modifiée.

Scénario 3 : Supprimer une relation de la liste de dépendance.
Pré condition : relation existe déjà dans la liste des dépendances.

Supprimer une relation de la liste des dépendances :
1-Accéder à l'onglet relation.
2-Choisir onglet ''la création''.
3-Sélectionner une relation.
4-cliquer sur supprimer.
Post condition : relation de dépendance supprimée.

Scénario 4 : Supprimer une relation de la liste des relations
Pré condition : relation existe déjà dans la liste des relations.
Supprimer une relation de la liste des relations :
1-Accéder à l'onglet liste des relations.
2-Sélectionner une relation.
5-cliquer sur supprimer.
Post condition : relation supprimée de la liste des relations.

- **Description textuelle: Gérer diagramme de déploiement étendu**

Titre : Gérer diagramme de déploiement.
Résumé : ce cas d'utilisation permet de gérer des acteurs, des nœuds, afficher des composants, et de gérer les connexions entre les intervenant du diagramme.
Acteur : utilisateur. **Responsable:**BaghdadiSihem
Pré condition : Application déjà sur papier.
Scénario nominal 1 : Gérer acteur**.**
Scénario 1 : Ajouter un acteur**.**
Ajouter un acteur.
1-Accéder à l'application DAC.
2-Afficher l'onglet du diagramme de déploiement.
3-Choisir onglet acteur.
4-choisir onglet Propriété.
5- saisir le nom.
6- cliquer sur ajouter.
Scénario 2 : modifier un acteur.
Pré condition : acteur existe déjà dans la liste.
Modifier un acteur:
1- Accéder à l'onglet acteur.
2-Afficher l'onglet liste des acteurs.
3-Sélectionner un acteur.
5-cliquer sur modifier.
6-Afficher l'onglet propriétés.
7-Saisir à nouveau le nom.
8- Cliquer sur valider.
Post condition : acteur modifiée.

Scénario 3 : Supprimer un acteur.
Pré condition : acteur existe déjà dans la liste.

Supprimer un acteur:
1-Accéder à l'onglet acteur.
2-Afficher l'onglet liste des acteurs.
3-Sélectionner un acteur.
5-cliquer sur supprimer.
Post condition : acteur supprimé.
Scénario d'erreur:
Scénario d'erreur : Nom vide
Etape : Affichage d'une boite de dialogue'' vous devez saisir le nom d'acteur svp''.
Etape : Retourner au scénario 1 étape5.
Scénario d'erreur : Nom dupliqué
Etape : Affichage d'une boite de dialogue "l'acteur existe déjà"
Etape : Retourner au scénario 1 étape 5.

Scénario nominal 2: Gérer nœud.
Scénario 1 : Ajouter un nœud.
1-Accéder à l'application DAC.
2-Afficher l'onglet du diagramme de déploiement.
3-Choisir onglet nœud.
4-choisir onglet Propriété.
5- saisir le nom.
6- cliquer sur ajouter
.**Scénario 2** : modifier un nœud.
Pré condition : nœud existe déjà dans la liste.
Modifier un nœud:
1- Accéder à l'onglet acteur.
2-Afficher l'onglet liste des nœuds.
3-Sélectionner un nœud.
5-cliquer sur modifier.
6-Afficher l'onglet propriétés.
7-Saisir à nouveau les propriétés.
8- Cliquer sur valider.
Post condition : nœud modifiée.

Scénario 3 : Supprimer un nœud.
Pré condition : nœud existe déjà dans la liste.
Supprimer un nœud :
1-Accéder à l'onglet nœud.
2-Afficher l'onglet liste des nœuds.
3-Sélectionner un nœud.
5-cliquer sur supprimer.
Post condition : nœud supprimé.
Scénario d'erreur:
Scénario d'erreur : champs vide

Etape : Affichage d'une boite de dialogue‘’ vous devez remplir tous les champs svp’’.
Etape : Retourner au scénario 1 étape 5.
Scénario d'erreur : Nœud dupliqué
Etape : Affichage d'une boite de dialogue "l'acteur existe déjà"
Etape : Retourner au scénario 1 étape 5.

- **Description textuelle : Gérer des connexions**

Titre : Gérer des connexions dans un diagramme de déploiement.
Résumé : ce cas d’utilisation permet d’ajouter, modifier et supprimer des connexions entre les différents intervenants du diagramme de déploiement.
Acteur : utilisateur. **Responsable**:BaghdadiSihem
Pré condition : diagramme de composant et de déploiement déjà géré.
Scénario nominal 3: Gérer connexion.
Scénario 1 : Ajouter une connexion.
1-Accéder à l’application DAC.
2-Afficher l’onglet du diagramme de déploiement.
3-Choisir onglet connexion.
4-Choisir onglet la création.
5-saisir le nom.
6-choisir acteur, composant ou nœud de départ.
7-choisir acteur, composant ou nœud d'arrivé.
8-saisir débit.
9-cliquer sur ajouter.
10-Générer les matrices.
Scénario alternatif :
Le système n’acceptera pas de laisser un champ vide il affiche ‘’svp vous devez remplir tous les champs’’.
Le système n’acceptera pas un nom écrit deux fois.
Post condition : connexion ajoutée dans la liste des connexions, affichage de connexions dans la liste de dépendance et la génération des matrices.

Scénario 2 : Modifier une connexion.
Pré condition : connexion existe déjà dans la liste des connexions.

Modifier une connexion de la liste des connexions:
1- Accéder à l’onglet liste des connexions.
2-Sélectionner une connexion.
5-cliquer sur modifier.
6-Afficher l’onglet connexion.
7-Saisir à nouveau les nouvelles propriétés sur l’onglet ‘’la création’’.
8- Cliquer sur valider.
Post condition : connexion modifiée.

Scénario 3 : Supprimer une connexion de la liste de dépendance.

Pré condition : connexion existe déjà dans la liste des dépendances.
Supprimer une connexion de la liste des dépendances :
1-Accéder à l'onglet connexion.
2-Choisir onglet ''la création''.
3-Sélectionner une connexion.
4-cliquer sur supprimer.
Post condition : connexion de dépendance supprimée.
Scénario 4 : Supprimer une connexion de la liste des connexions
Pré condition : connexion existe déjà dans la liste des connexions.
Supprimer une connexion de la liste des connexions :
1-Accéder à l'onglet liste des connexions.
2-Sélectionner une connexion.
5-cliquer sur supprimer.
Post condition : connexion supprimée de la liste des connexions.

- **Description textuelle : Gérer politique de sécurité.**

Titre : Gérer politique de sécurité.
Résumé : ce cas d'utilisation permet d'autoriser ou d'interdire l'accès et de calculer leurs matrices.
Acteur : utilisateur. **Responsable**: Baghdadi Sihem
Pré condition : diagramme de composant et de déploiement déjà gérer.
Scénario nominal 1 : Permettre l'accès.
Ajouter.
Scénario 1: Ajouter une connexion et calculer matrice d'autorisation.
1-Accéder à l'application DAC.
2-Afficher l'onglet politique de sécurité.
3-Choisir autorisé.
4-Sélectionner acteur.
5- Sélectionner composant
6-Cliquer sur ajouter.
7-cliquer sur valider (calculer matrice d'autorisation).

Scénario 2 : Modifier la connexion permission d'accès.
Pré condition : Connexion existe déjà entre acteur composant.
Modifier:
1- Accéder à l'onglet autorisé.
2-Sélectionner connexion.
3-cliquer sur modifier.
5-cliquer sur modifier.
6-faire la sélection de nouveau.
7- Cliquer sur valider (Recalculer matrice d'autorisation).
Post condition : connexion modifiée.

Scénario 3 : Supprimer la connexion de la permission d'accès.
Pré condition : connexion entre acteur et composant existe déjà.
Supprimer:
1-Accéder à l'onglet autorisé.
2-Sélectionner une connexion.
3-cliquer sur supprimer.
Post condition : connexion supprimé.
Scénario d'erreur : Aucun.
Scénario Alternatif : Aucun.

Scénario nominal 2 : Interdire l'accès.
Scénario 1: Ajouter une connexion et calculer matrice d'interdiction
1-Accéder à l'application DAC.
2-Afficher l'onglet politique de sécurité.
3-Choisir interdit.
4-Sélectionner acteur.
5- Sélectionner composant
6-Cliquer sur ajouter.
7-cliquer sur valider (calculer matrice d'interdiction pour l'accès).

Scénario 2 : Modifier la connexion d'interdiction pour l'accès.
Pré condition : Connexion existe déjà entre acteur composant.
Modifier:
1- Accéder à l'onglet interdit.
2-Sélectionner connexion.
3-cliquer sur modifier.
5-cliquer sur modifier.
6-faire la sélection de nouveau.
7- Cliquer sur valider (recalculer matrice d'interdiction pour l'accès).
Post condition : connexion modifiée.

Scénario 3 : Supprimer la connexion d'interdiction pour l'accès.
Pré condition : connexion entre acteur et composant existe déjà.
Supprimer:
1-Accéder à l'onglet interdit.
2-Sélectionner une connexion.
3-cliquer sur supprimer.
Post condition : connexion supprimé.
Scénario d'erreur : Aucun.
Scénario Alternatif : Aucun.

III.8.2.2. Diagramme de classe d'analyse :

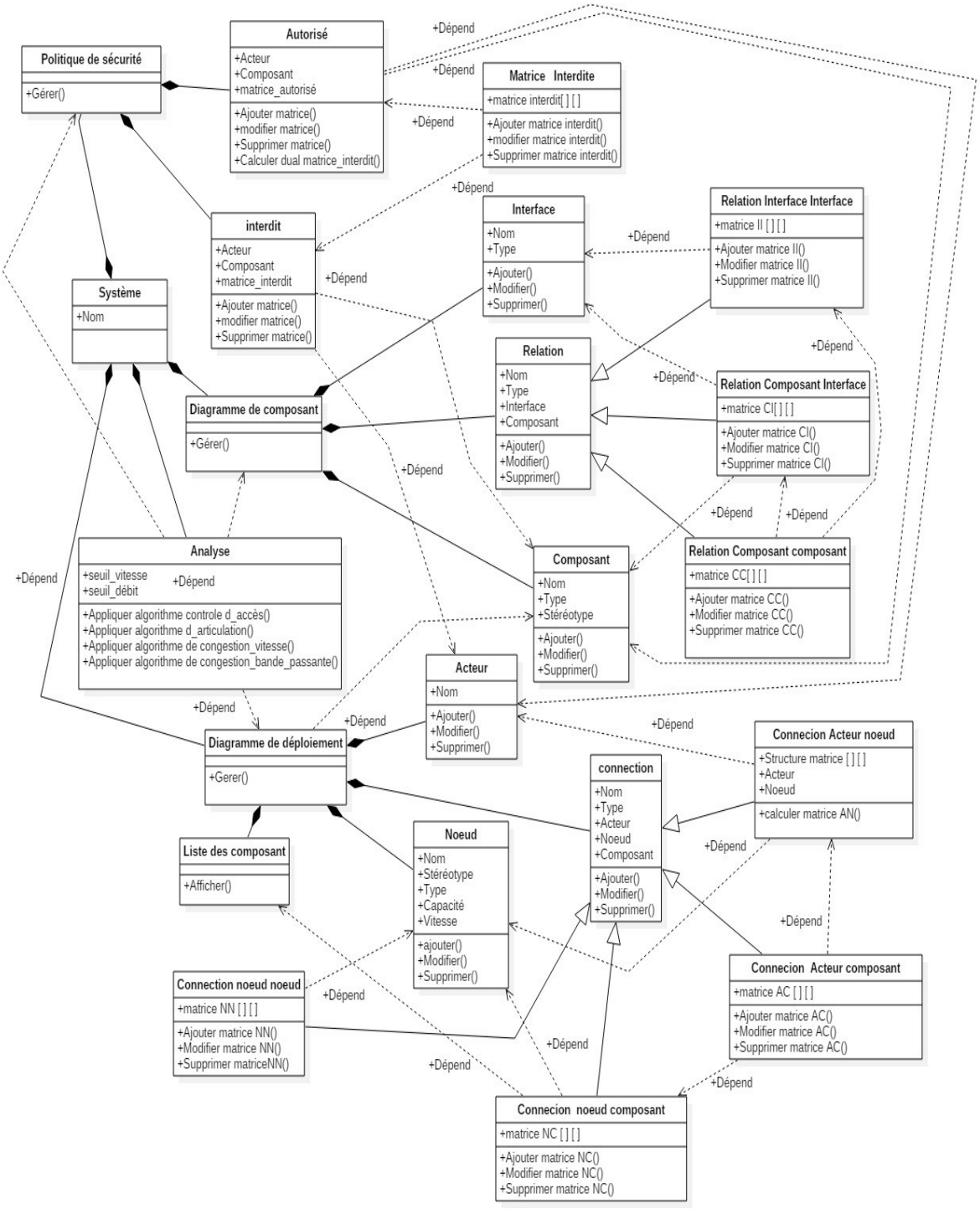

Figure III.11. Diagramme de classe d'analyse de l'application

III.8.3. Conception

III.8.3.1. Diagramme de classe de conception

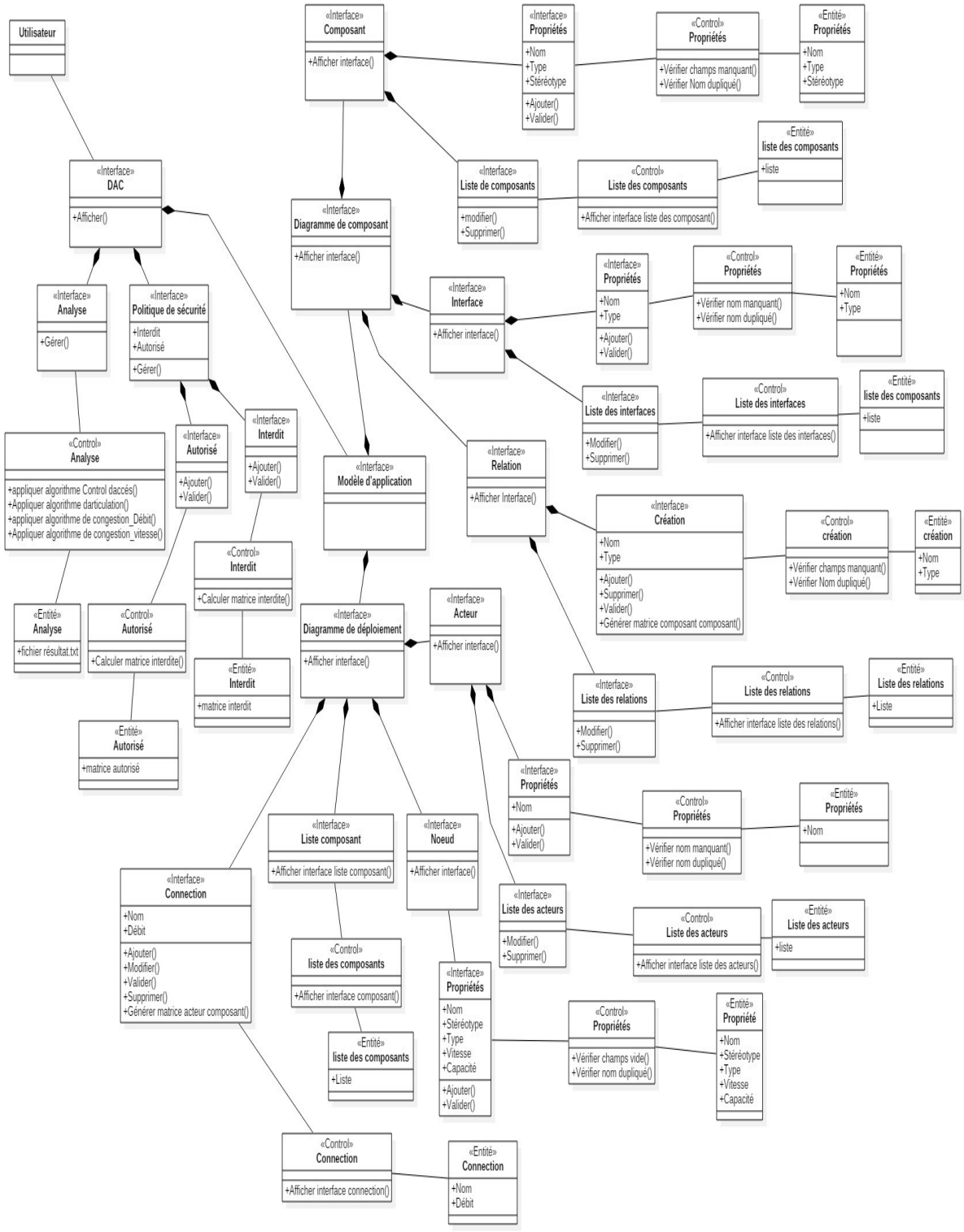

Figure **III.12. Diagramme de classe de conception de l'application**

III.8.3.2. Diagramme de séquence de conception de l'application

III.8.3.2.1. Diagramme de séquence (diagramme de composant : gérer composant)

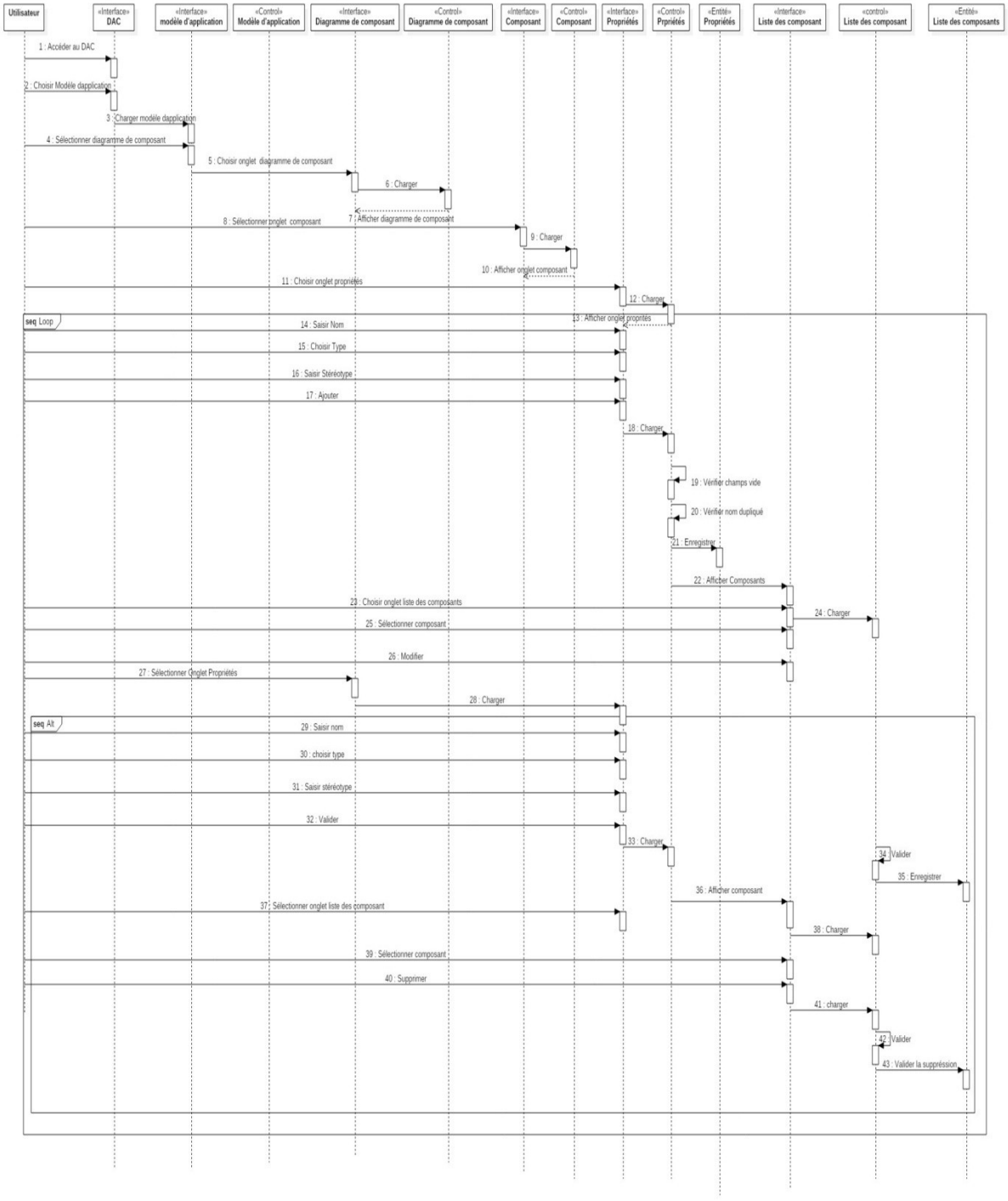

Figure III.13. Diagramme de séquence de modèle d'application (diagramme de composant : Gérer Composant)

III.8.3.2.2. Diagramme de séquence (diagramme de composant : gérer interface)

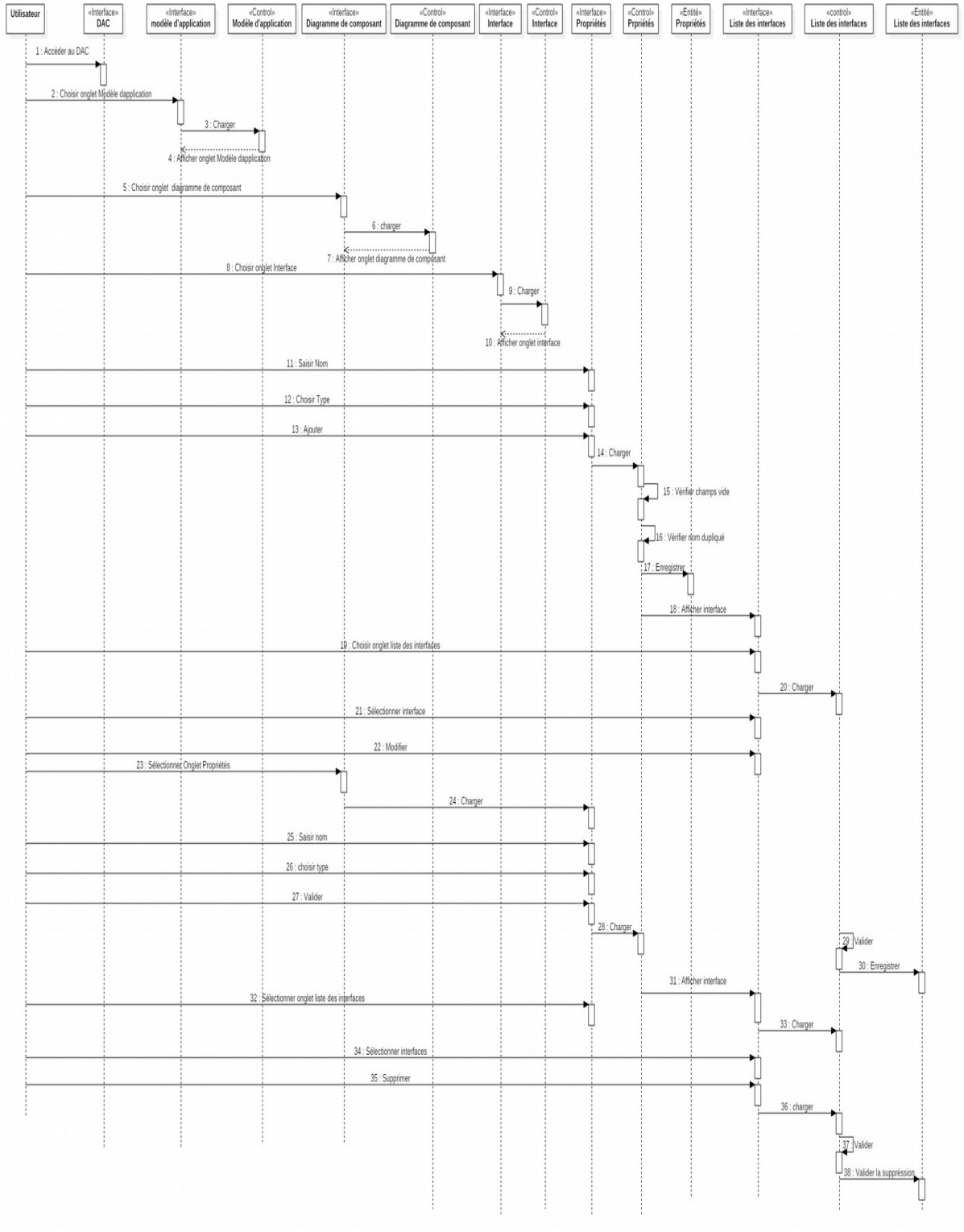

Figure III.14 Diagramme de séquence de modèle d'application (diagramme de composant : Gérer Interface).

III.8.3.2.3. Diagramme de séquence (diagramme de composant : gérer relation)

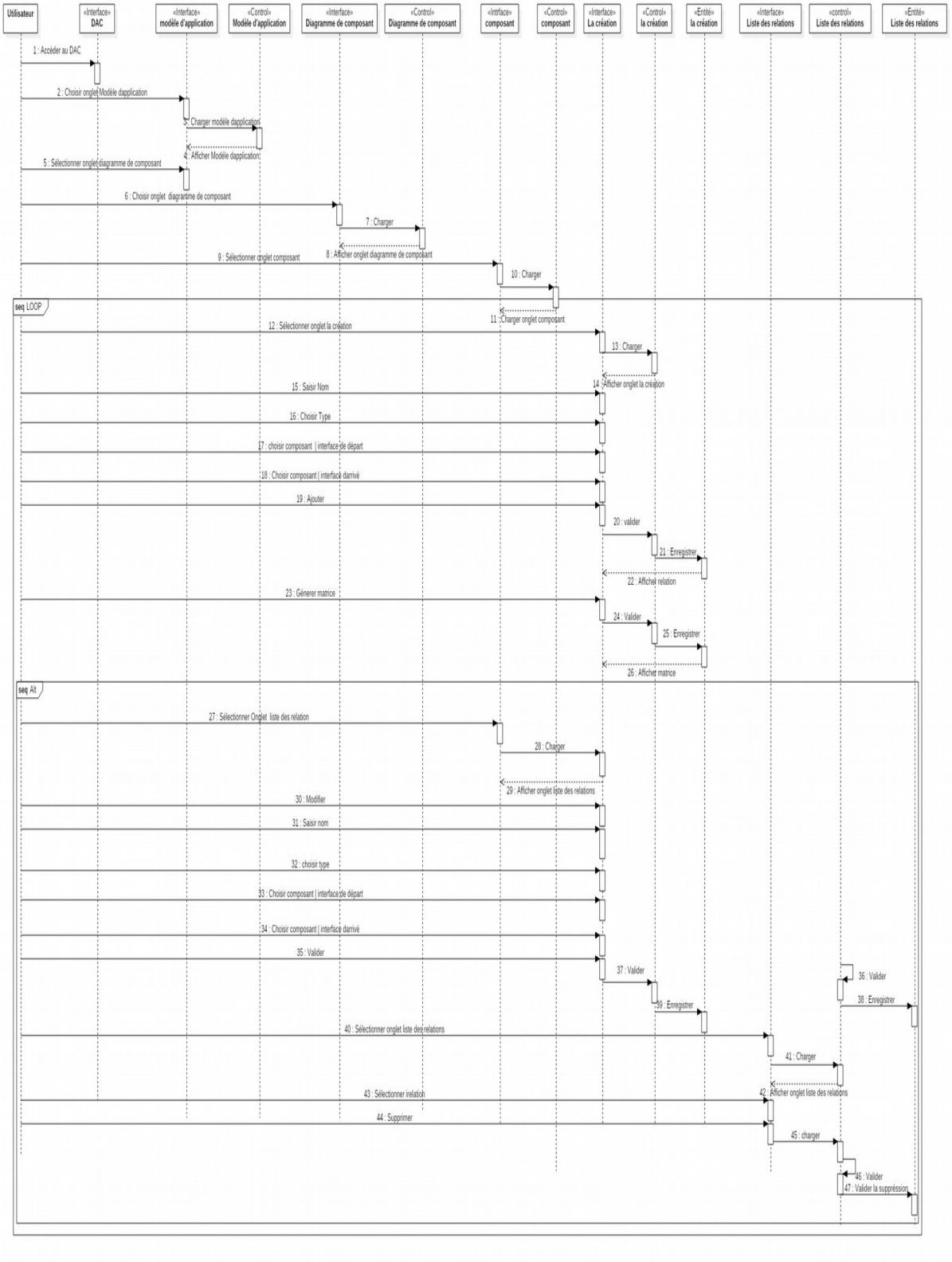

Figure III.15. Diagramme de séquence de modèle d'application (Diagramme de composant : Gérer relation)

III.8.3.2.4. Diagramme de séquence (diagramme de déploiement étendu: gérer acteur)

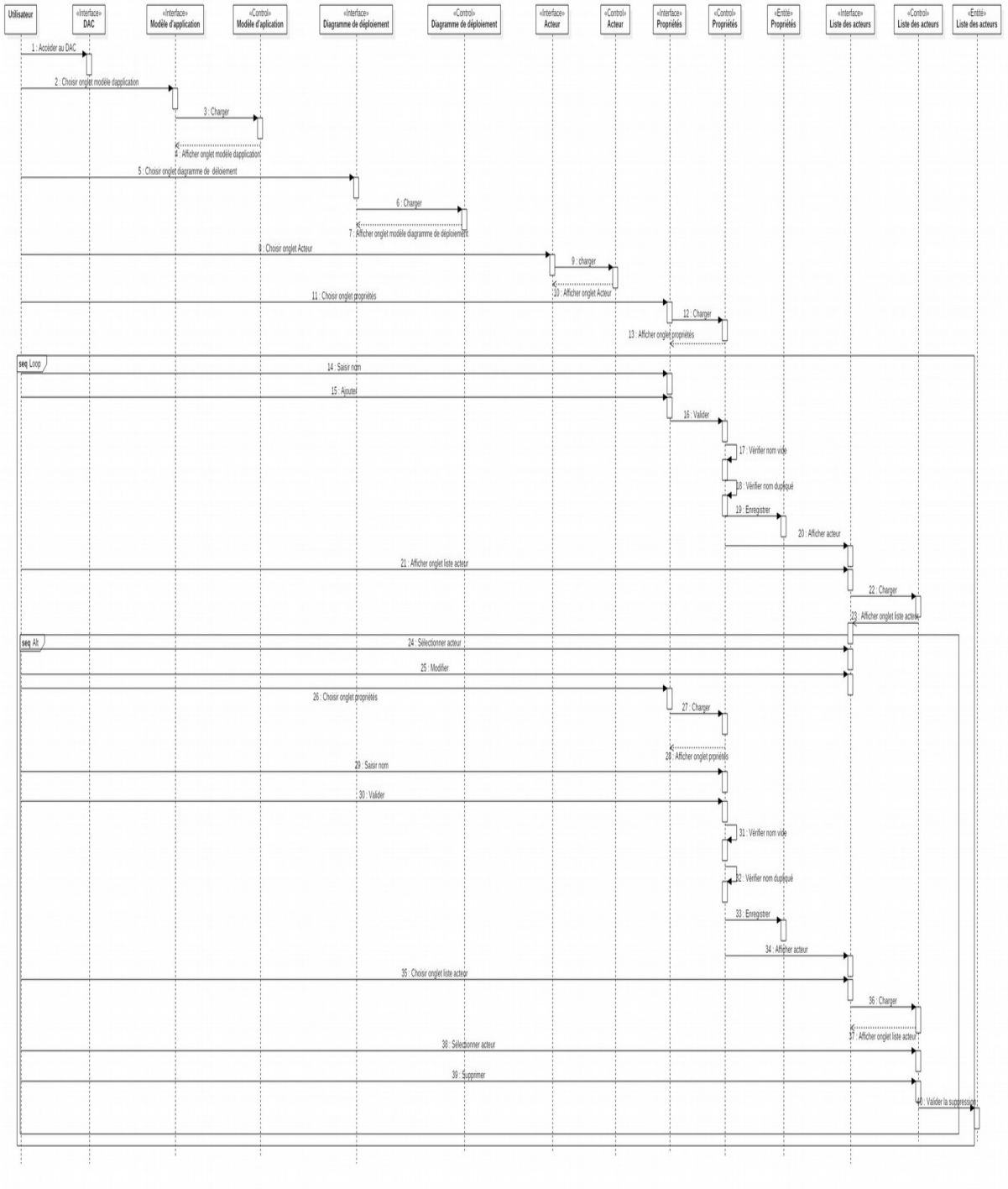

Figure III.16. Diagramme de séquence de modèle d'application (Diagramme de déploiement étendu (Gérer acteur))

III.8.3.2.5. Diagramme de séquence (diagramme de déploiement étendu: gérer nœud)

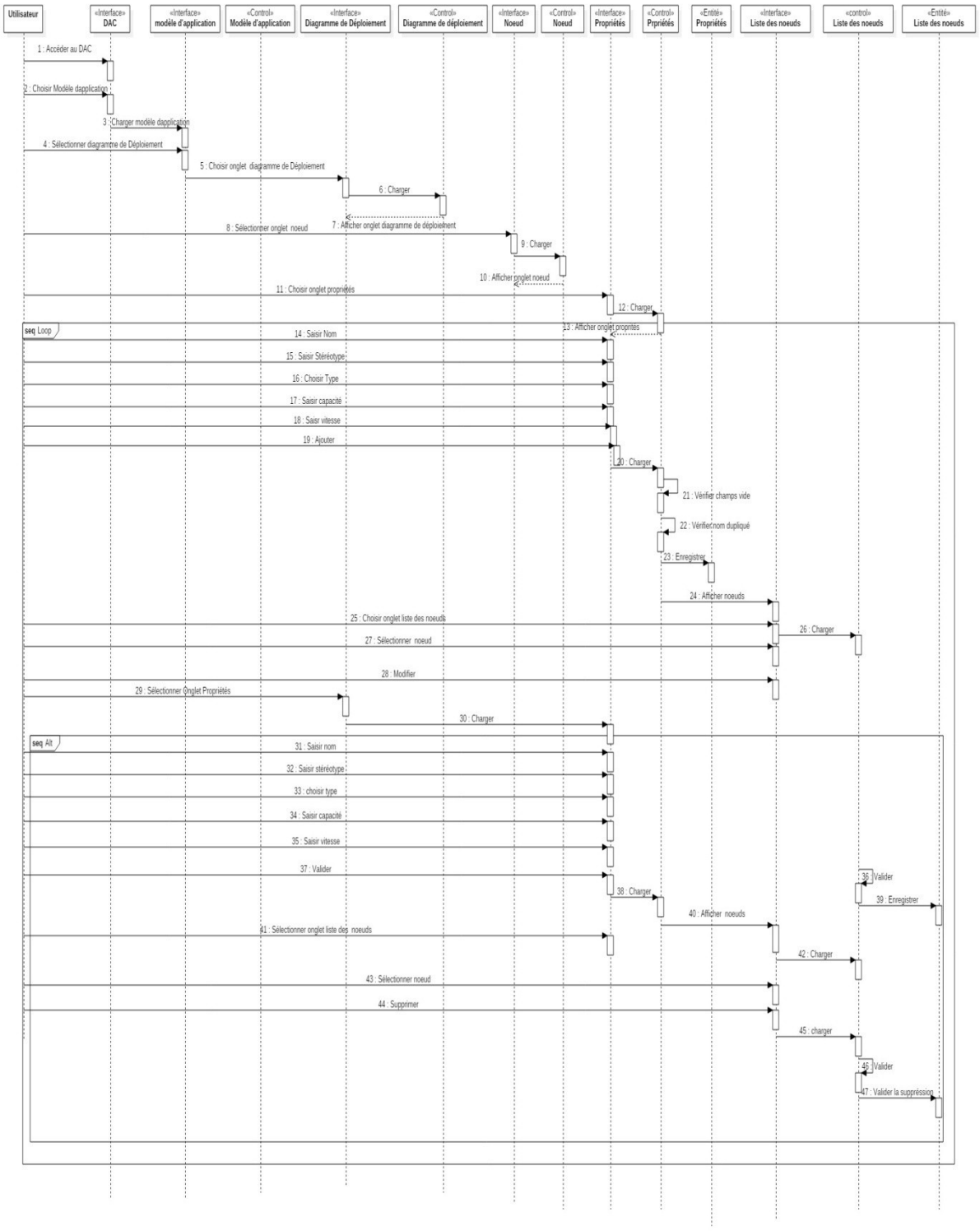

Figure III.17. Diagramme de séquence de modèle d'application (Diagramme de déploiement étendu : Gérer nœud)

III.8.3.2.6. Diagramme de séquence (diagramme de déploiement étendu : gérer connexions)

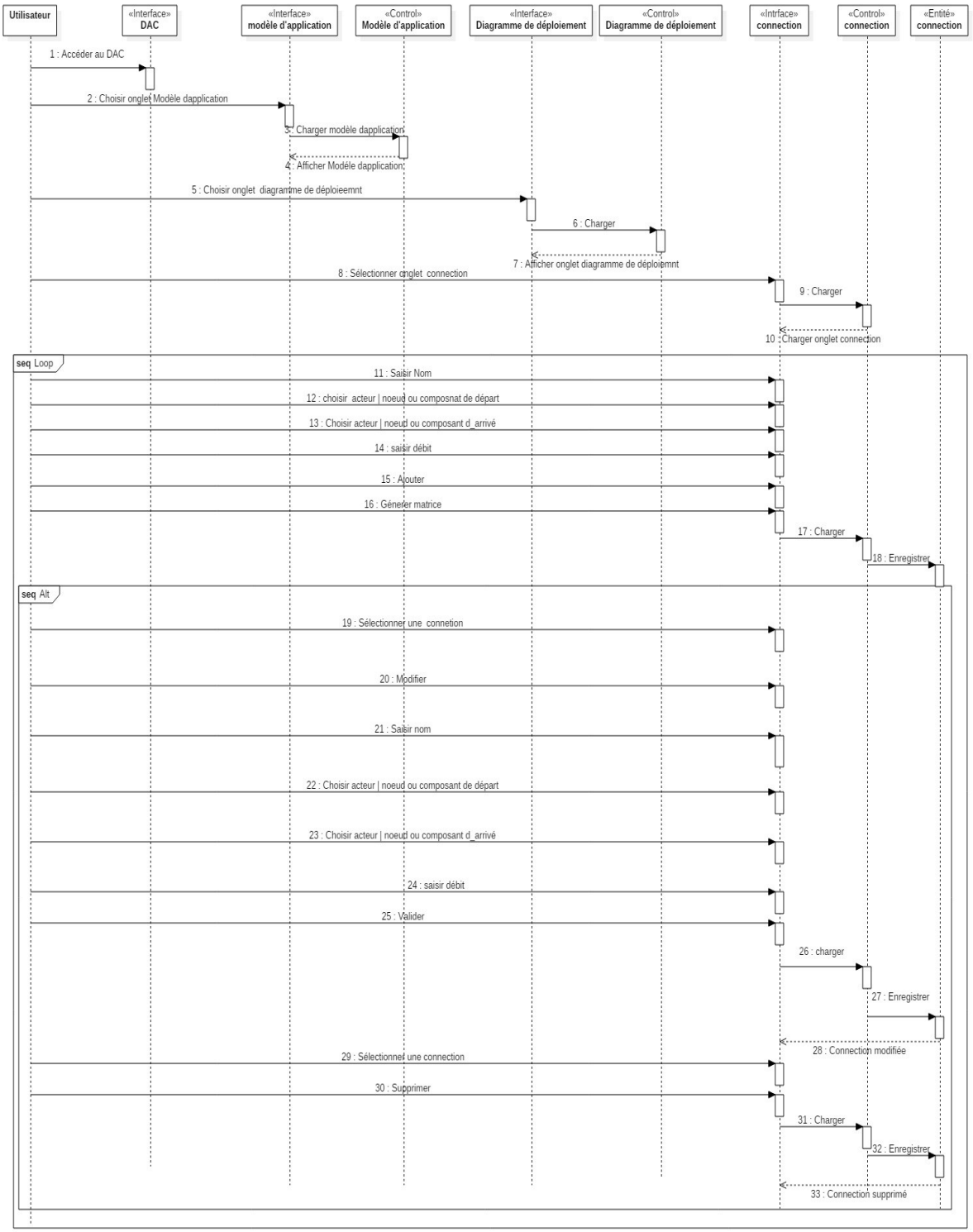

Figure III.18. Diagramme de séquence de modèle d'application (diagramme de déploiement étendu : gérer connexions)

III.8.3.2.7. Diagramme de séquence (gérer politique de sécurité)

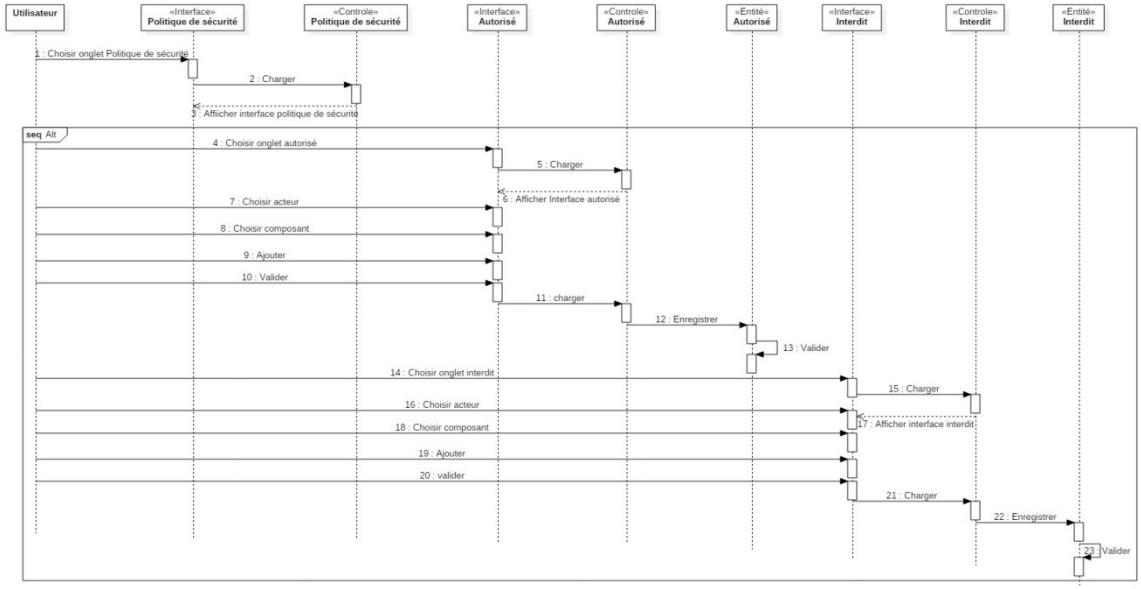

Figure III.19. Diagramme de séquence (Gérer politique de sécurité).

III.8.3.2.8. Diagramme de séquence (gérer analyse)

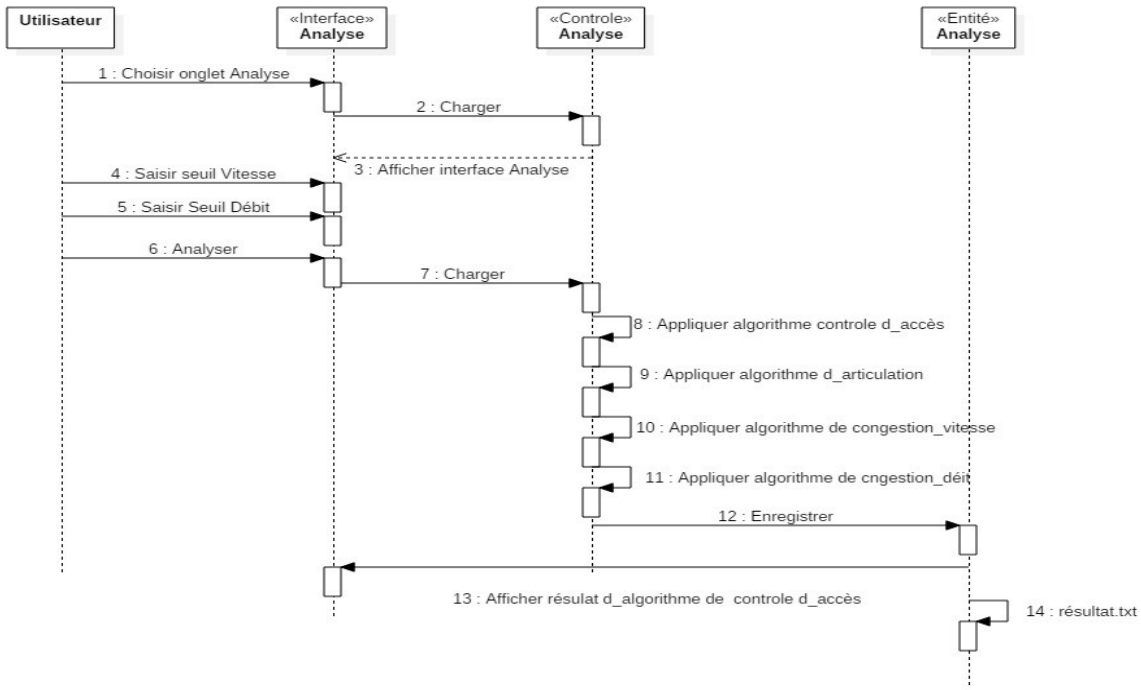

Figure III.20. Diagramme de séquence (gérer analyse).

III.8.3.3. Diagramme de composant de l'application :

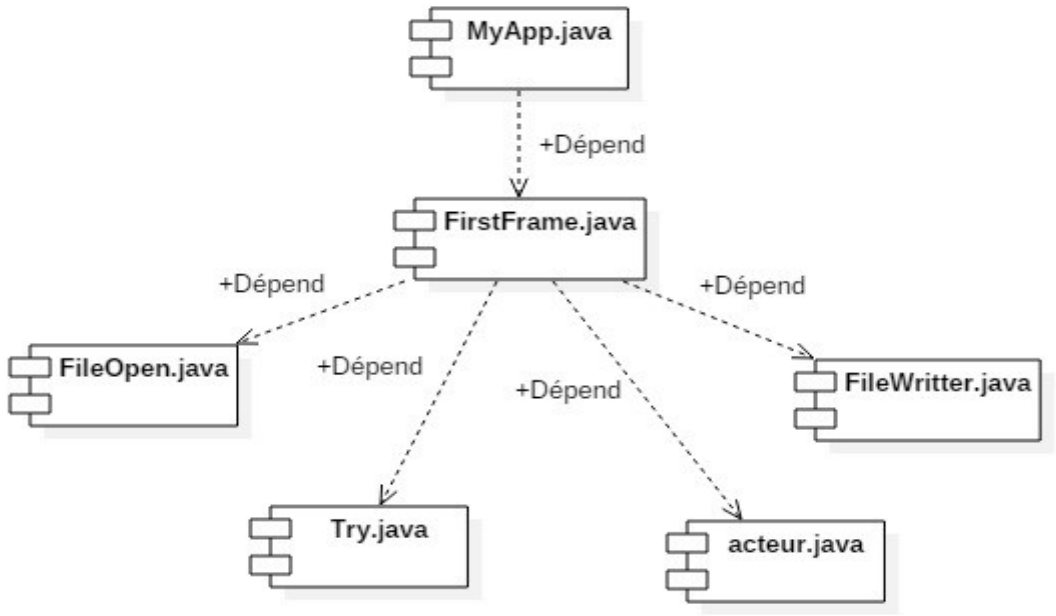

Figure III.21. Diagramme de composant de l'application.

III.8.3.4. Diagramme de déploiement de l'application :

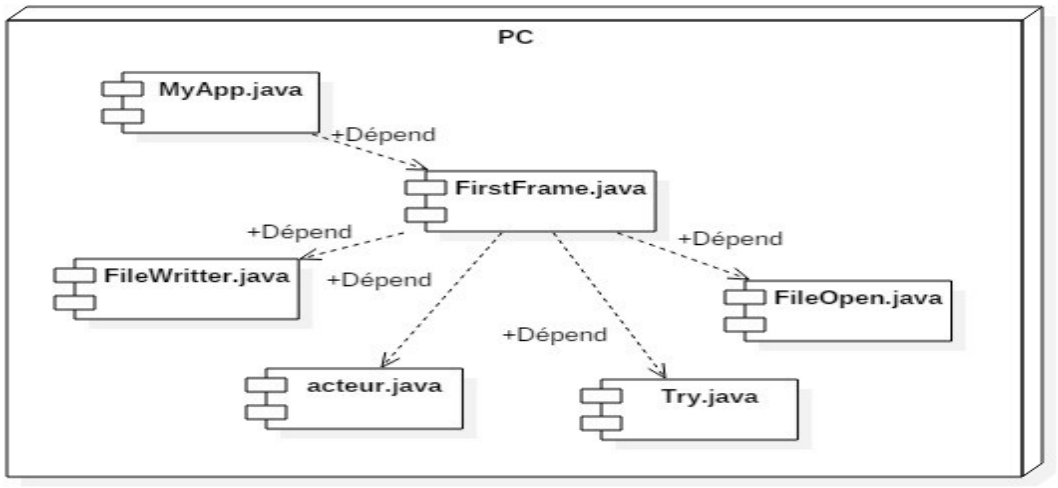

Figure III.22. Diagramme de déploiement de l'application.

III.9. Conclusion

Dans ce chapitre nous avons décrit la partie théorique de notre application, nous avons détaillé les algorithmes utilisés et modélisé notre application, nous allons nous appuyer sur ce chapitre pour implémenter notre application.

IV.1. Introduction

Nous abordons dans ce chapitre la partie implémentation, qui mettra en pratique la conception de notre modèle de contrôle d'accès DAC Pour ce faire nous présentons d'abord le langage de programmation et l'environnement de développement, puis nous passons à la présentation des interfaces de notre application et expliquer son fonctionnement. Nous clôturons ce chapitre par un ensemble de test d'évaluation.

IV.2. Langage de programmation (JAVA)

L'application est développée en utilisant le langage de programmation JAVA. Java est un langage de programmation objet et un environnement d'exécution récent, développé par Sun Microsystems en 1991.

IV.2.1. Pourquoi avons-nous choisi le langage JAVA ?

Java possède de nombreuses caractéristiques qui font de lui un langage de choix. En effet :

- Java est un langage de programmation parmi les plus utilisés au monde, donc on a plus de chance d'avoir des solutions pour nos problèmes sur le Net, car java est le langage de programmation le plus documenté.
- Les programmes développés sous Java sont portables, c'est-à-dire on peut les exécuter presque sur n'importe quel système d'exploitation, sauf pour certains cas où on utilise des API non compatibles avec certains OS.
- Java met à disposition plusieurs niveaux de protection de données (les modes : public, private, ...), ainsi qu'un mécanisme de traitement très sophistiqué des exceptions.
- Java a des mécanismes permettant au programmeur de faire de la programmation concurrente (multi-thread) [29].

Pour l'implémentation du modèle, nous avons exploité la plateforme Netbeans.

IV.3. Environnement de développement

NetBeans est un projet open source ayant un succès et une base d'utilisateur très large, une communauté en croissance constante, et près 100 partenaires mondiaux et des centaines de milliers d'utilisateur à travers le monde. Sun Microsystems a fondé le projet open source NetBeans en Juin 2000 et continue d'être le sponsor principal du projet.

Aujourd'hui, deux projets existent: L'EDI NetBeans et la Plateforme NetBeans.

- **L'EDI NetBeans :** est un environnement de développement - un outil pour les programmeurs pour écrire, compiler, déboguer et déployer des programmes. Il est écrit en Java - mais peut supporter n'importe quel langage de programmation. Il y a également un grand nombre de modules pour étendre l'EDI NetBeans. L'EDI NetBeans est un produit gratuit, sans aucune restriction quant à son usage.

- **La Plateforme NetBeans :** qui est une fondation modulable et extensible utilisée comme brique logicielle pour la création d'applications bureautiques. Les partenaires privilégiés fournissent des modules à valeurs rajoutées qui s'intègrent facilement à la Plateforme et peuvent être utilisés pour développer ses propres outils et solutions [30].

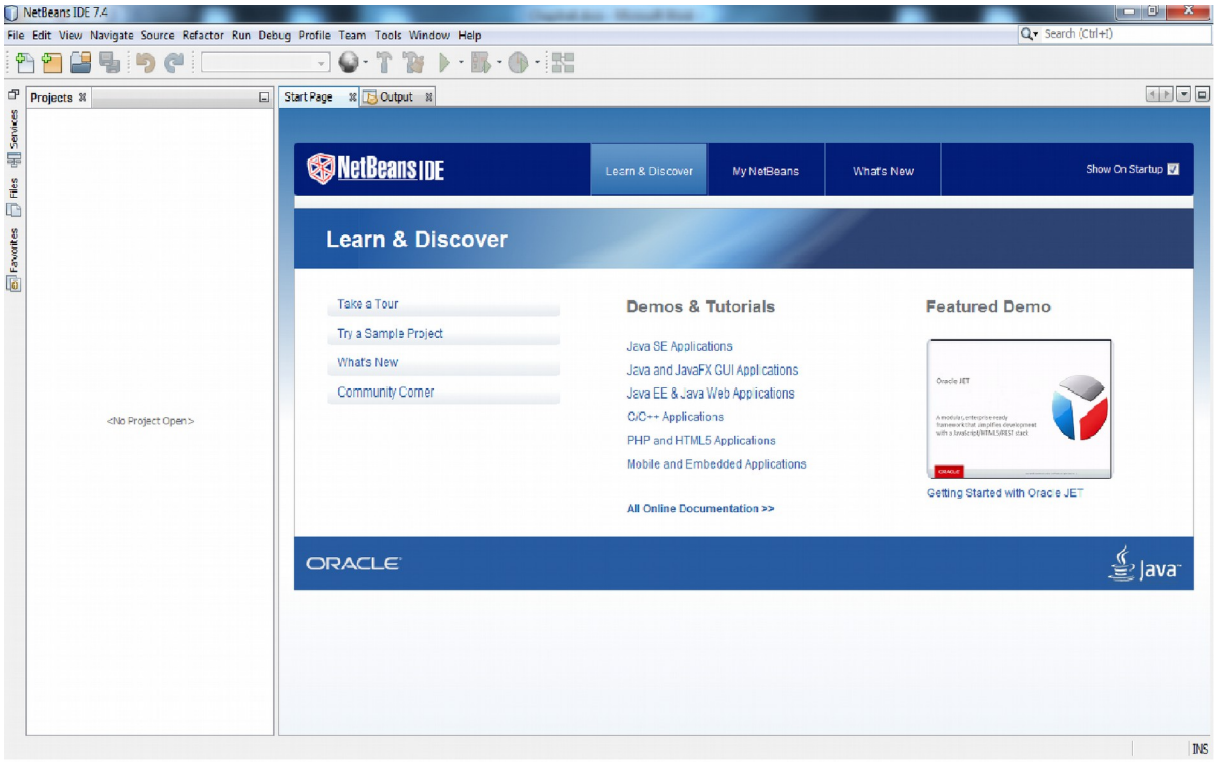

Figure IV.1. Interface de l'IDE NetBeans.

IV.4. Code source de quelques fonctions utilisées dans l'application

Dans ce qui va suivre nous allons proposer code source en java de quelque algorithmes décrit dans le chapitre précédent. Les tableaux 12, 13,14 montrent les fragments de code qui implémentent les algorithmes de la recherche du point d'articulation, de congestion vitesse et de congestion débit respectivement.

Tableau IV.12. Fonction Articulation

```
public void Articulation()
  {int nb=0;
  int maxR=2;
      for (int k = 0; k < Noeud.size(); k++) {
         nb=0;
       for (int l = 0; l< Noeud.size(); l++) {
          if(NN[k][l]==1)
             nb++;
       }
       if(maxR<=nb)
          Ana.addElement("le noeud"+K+"est un point d'articulation");
      }
  }
```

Tableau IV.13. Fonction Congestion_vitesse.

```
public void Congestion_Vitesse()
   {String rep="Aucune faille n'a été Détécte";
      for (int k = 0; k < Noeud.size(); k++) {
        for (int l = 0; l< Noeud.size(); l++) {
          if(NN[k][l]==1)
          {
             if(seuil<valeurA(Nvitesse.get(k)-Nvitesse.get(l)))
                rep="Une faille n'a été Détécte";
          }

        }
      }
      write("le resultat de l'algorithme de congestion_Vitesse"+rep);
   }
```

Tableau IV.14. Fonction Congestion_débit.

```
public void Congestion_débit()
    {String res="Aucune faille n'a été détécté";
    double prec=0;
      for (int k = 0; k < Noeud.size(); k++)
         for (int l = 0; l < Noeud.size(); l++) {
           System.out.println("val"+NN[k][l]);
           if(NN[k][l]!=0)
           {

              if((prec-NNd[k][l])< seuilD)
                 res="Une faille n'a été détécté";
              prec=NNd[k][l];
           }                                  }
      write(
"le resultats de l'algorithme de congestion_débit----> "+res);
```

```
}
```

IV.5. Implémentation et teste de l'application

Notre application est composée de deux parties, la première partie consiste à introduire le modèle d'application (les diagrammes UML) et la politique de contrôle d'accès 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 (composant, interface, nœud, acteur …..) et un autre module pour la détection des défiances de sécurité du modèle d'application.

Pour le teste nous avons choisi l'exemple décrit dans le chapitre précédentles resultat sont affichés dans le fichier résultat.

IV.5.1 Présentation des interfaces de notre application

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

IV.5.1.1 Interface de diagramme de composant

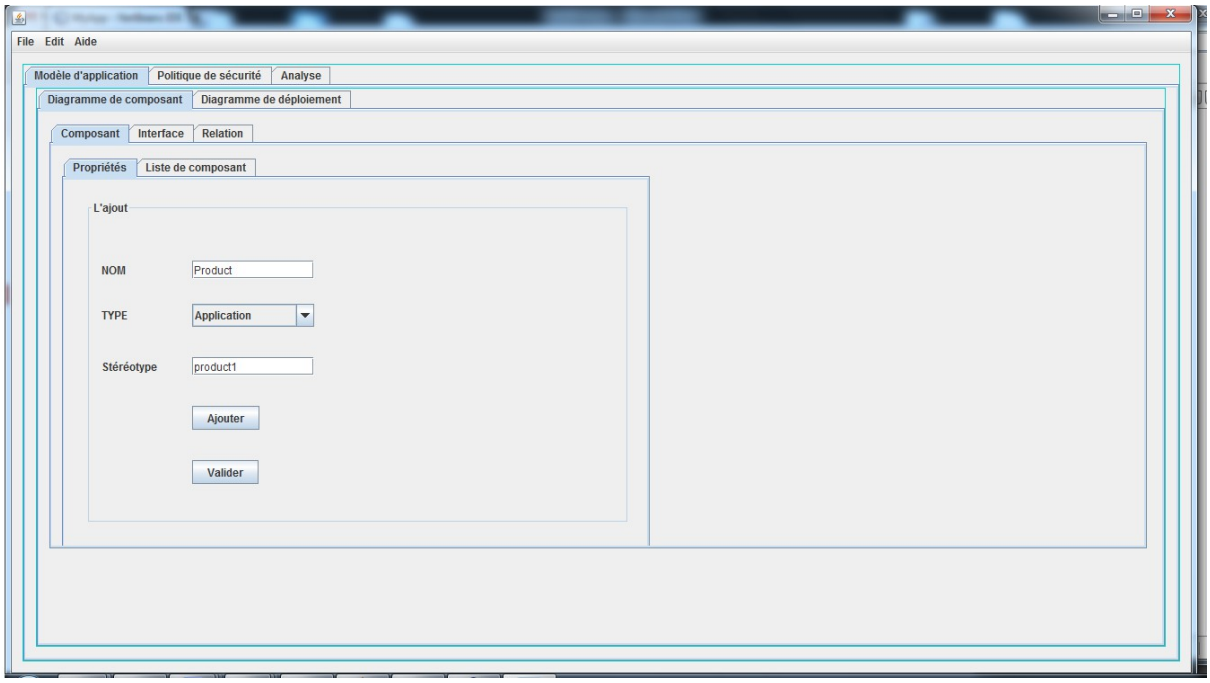

Figure IV.2. Interface modèle d'application (Gérer diagramme de composant).

IV.5.1.2. Interface de gérer relation

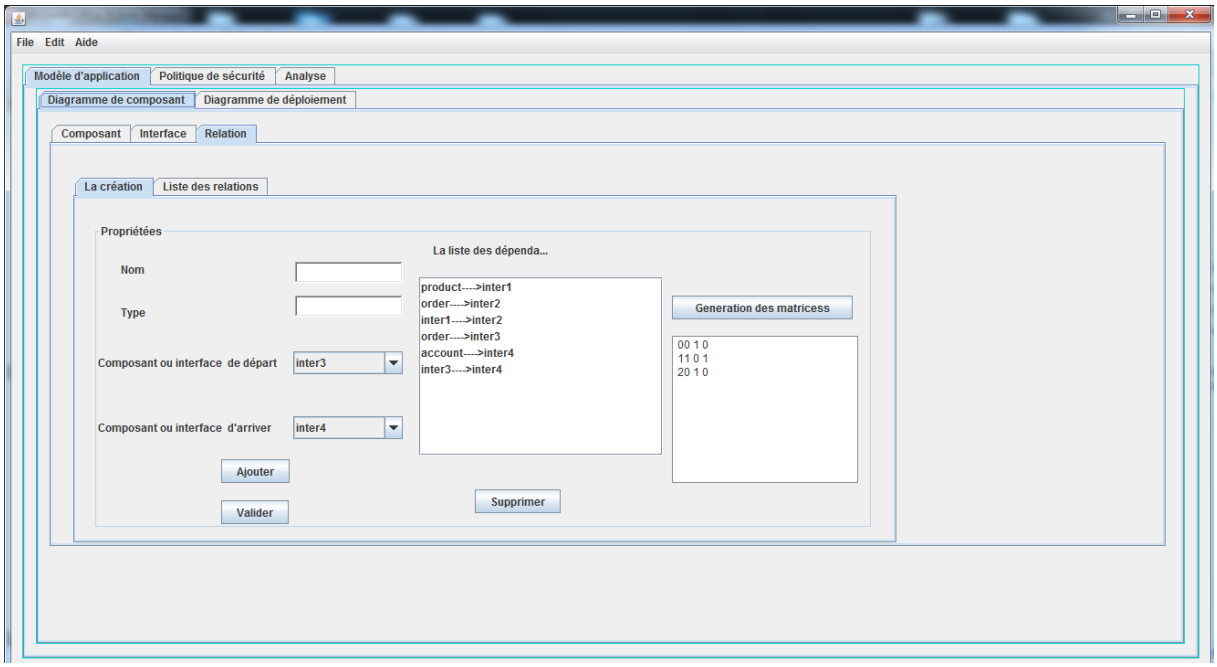

Figure IV.3. Interface modèle d'application (Gérer relations)

IV.5.1.3. Interface de la gestion de la politique de sécurité

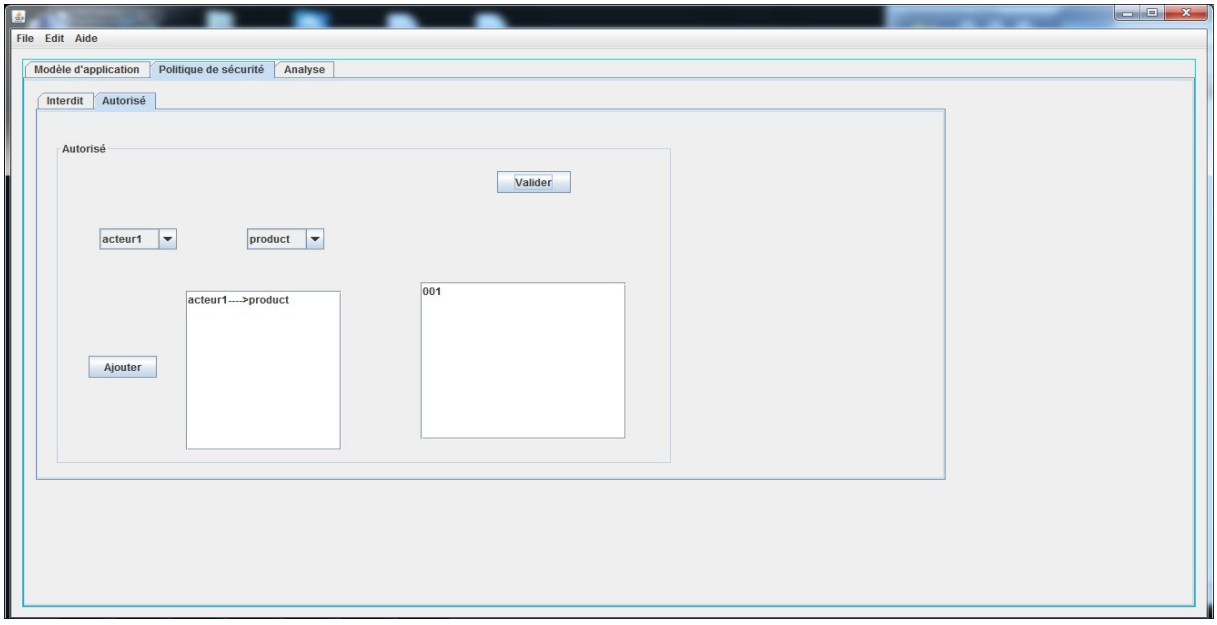

Figure IV.4. Interface Politique de sécurité (Gérer l'accès autorisé)

V.5.1.4. Interface de politique de sécurité cas interdit

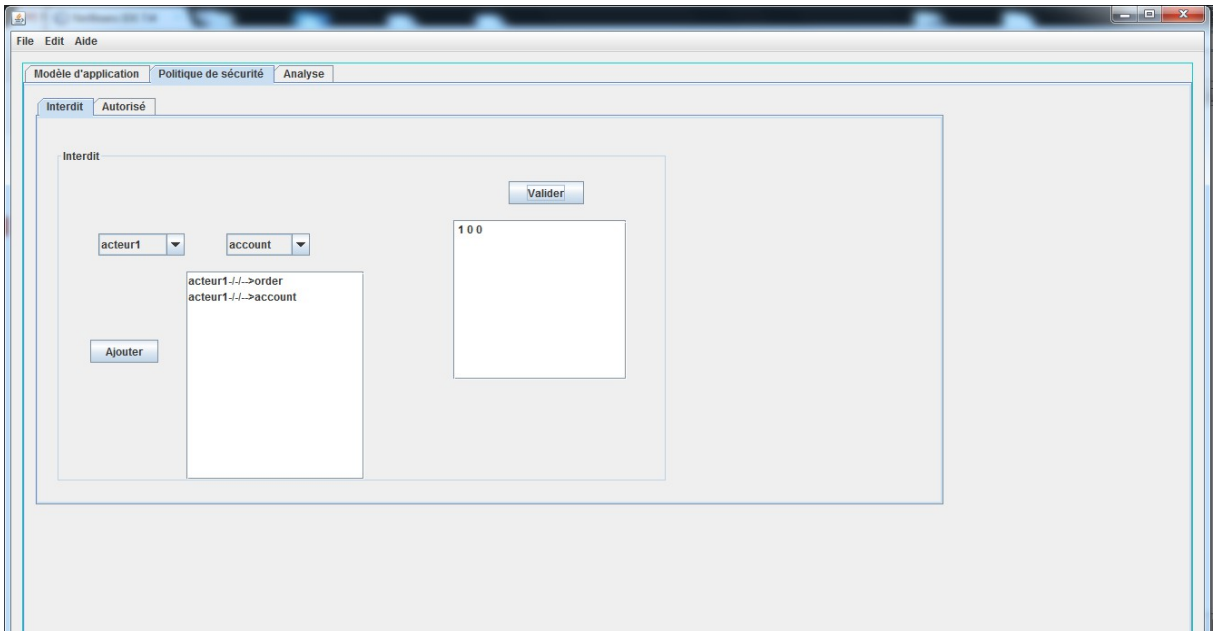

Figure IV.5. Interface politique de sécurité (Gérer l'accès interdit).

IV.5.1.5. Interface analyser

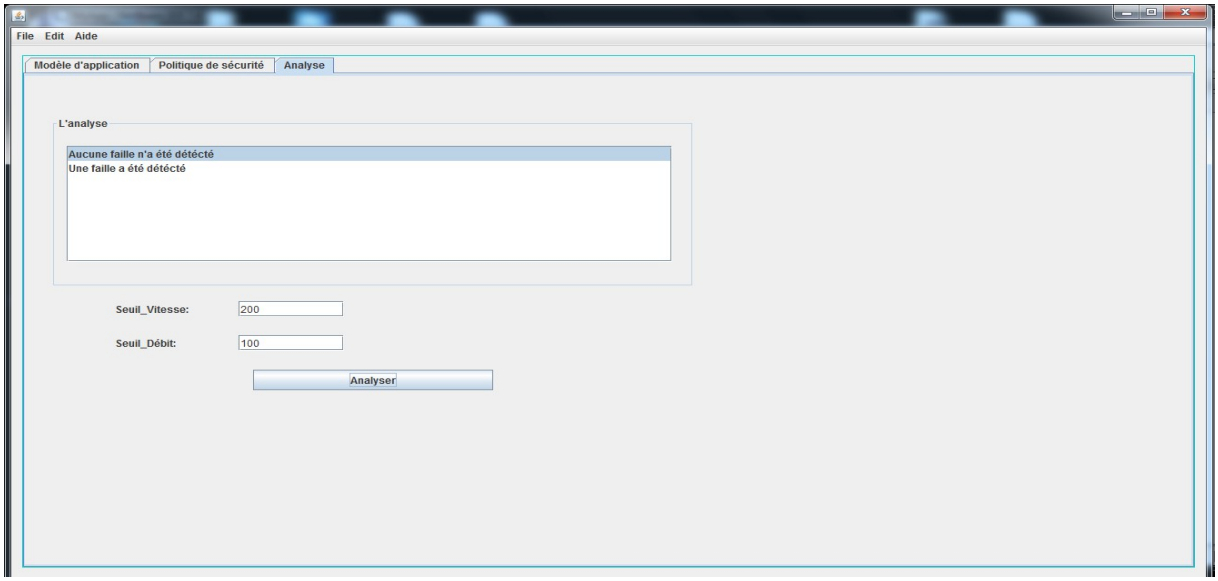

Figure IV.6. Interface d'analyse.

IV.5.1.6. Interface résultat

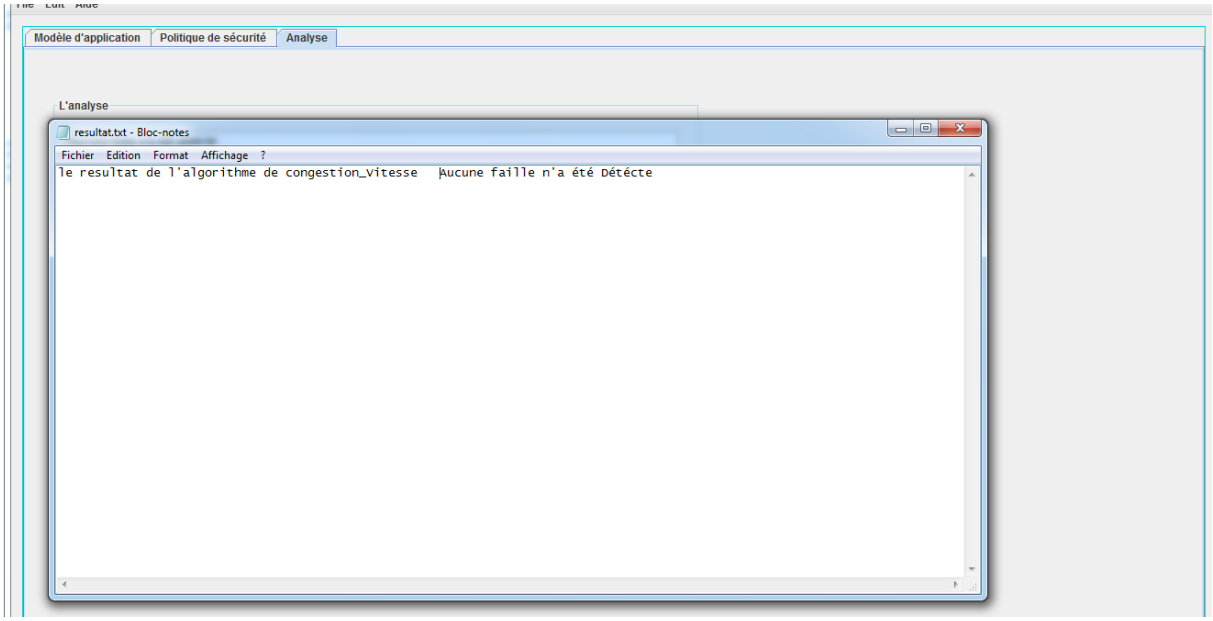

Figure IV.7. Interface d'analyse résultat de l'algorithme de congestion_vitesse.

IV.6. Conclusion

Dans ce chapitre nous avons commencé par définir le langage java suivi d'implémentation de quelque algorithme. Et enfin un teste (détaillé dans le chapitre précédent) est exécuté sur notre application (illustre par des interfaces de notre application).

Conclusion générale

Ce travail permet de déceler les problèmes de sécurité plus particulièrement la détection des failles avant de passer à la phase d'implémentation, c'est-à-dire la phase de conception ou modélisation, ce qui résulte un gain important de temps et d'argent.

Pour arriver à notre but nous somme passé par 3 grandes phases, la première phase concerne étude bibliographique (mini projet), la deuxième phase concerne analyse et conception de notre application et la troisième phase implémente la partie conception.

A fin d'atteindre notre objectif (application facile à utiliser et efficace à détecter les failles), nous avons proposé à l'utilisateur le moyen d'introduire les modèles d'application et une politique de sécurité en se basant sur le modèle DAC, ces données sont synthétisées dans des matrices adéquates, ces dernières seront interprétées comme des graphes (sommet, arc) sur les quels on a proposé des algorithmes qui nous ont permis de détecter les failles.

Toute fois ce travail ne permet pas de détecter toutes les failles décrite dans le chapitre trois mais les failles les plus rencontrés (problème de contrôle d'accès, d'articulation, de congestion (Vitesse, débit)).

Comme perspective nous suggérons :

- Etendre notre projet pour qu'il propose des corrections au système futur.
- Etendre la détection des failles en utilisant d'autre diagramme UML (activité, objet)
- Utiliser d'autre politique de sécurité tels que le MAC et RBAC pour spécifier la politique de sécurité.

Bibliographie

[1] Rumbaugh J., Jacobson I., Booch G., The Unified Modeling Language Reference Manual, Addison Wesley Longman, 0-201-30998-X, 1999 et 2004

[2] Roques P., UML 2 par la pratique Etudes de cas et exercices corrigés, Eyrolles, 2-212-12014-1, 2006.

[3] http://www.uml-diagrams.org. 2016

[4] P.S.Kaliappan, H.Koenig, V.K.Kaliappan, An Approach to Verify Component-based Designs 978-1- 4799-1823-2/15/ 2015 IEEE.

[5] V.Chourey , Dr. M. Sharma, Component Based Reliability Assessment from UML Models,978-1-4799-8792-4/15/ 2015 IEEE

[6] Object Management Group OMG. Uinified modeling language specification version 2.0 : Superstructure. Technical Report pct/03-08-02, OMG, 2003.

[7] Object Management OMG. Uinified modeling language specification version 2.0 : Infrastructure. Technical Report ptc/03-09-15, OMG, 2003.

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

[9] Computer Security Handbook, Seymour Bosworth, Michel E. Kabay (editors), John Wiley, 2002

[10] Anonyme, Sécurité maximal des systèmes et réseaux, CompusPress, 2003.

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

[12] S.P.Jajodia. S, “data security”, Research and practice I S 20(7): 537-556(95).

[13] 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.

[14] C P. Pfleeger and Shari Lawrence Pfleeger. Security in Computing (4th Edition). Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

[15] Vimercati.S, Samarat.P, "Access Control : Policies, Models, and Mechanisms", Foundations of Security Analysis and Design FOSAD, 2000.

[16] B. Lampson. "Protection" 5th Princeton Symposium on Information Sciences and Systems. 1971.

[17] M. Harrison, W. Ruzzo, J. Ullman. "Protection in operating systems". Communications of the ACM. 461–471. 1976.

[18] L. Lapadulaand D. Elliott Bell. Secure computer systems: A mathematical model.technical report mtr 2547 vol. 1, mitre corporation, bedford, massachusetts., 1973.

[19] J.R.CORNABAS THÈSE , "Formalisation de propriétés de sécurité pour la protection des systèmes d'exploitation ". 201

[20] S.Boulares, "Validation des politiques de sécurité par rapport aux modèles de contrôle d'accès". Université du Québec en Outaouais. 2010.

[21] F..D., Sandhu.R., and Gavrila. S,"Proposed NIST Standard for role-Based Access Control",ACM Transactions on Information and System Security, Vol, 4, 2001.

[22] H.Vijayakumar and J.Schiffman and T.Jaeger, Process Firewalls: Pretecting Processers During Resource Access,ACM 918-1-4503-1994-2/13/04,April 15-17,2013.

[23] O.Awodele, E .E.Onuiri, and S.O. Okolie, Vulnerabilities in Network Infrastructures and Prevention/Containment Measures, Proceedings of Informing Science & IT Education Conference (InSITE) 2012 .

[24] J.A.Wang and M.Goo,Vulnerability categorization Using Bayesian Network, ACM 978-1-4503-0017-9, 2010. Vulnérabilité

[25] Z.Feng and H.Wu and J.Su,Exploing Potential Vulnerabilities in Data Center Network, ACM 978-1-4503-0468-9/10/11, November /30/2010.

[26] V.H.Nguyer and le M.S.Tran ,Predicting vulnerable Software Components With Dependency Graphs,ACM 978-1-4503-0340-8, September 15/2010.

[27] http://www.academia.edu/4070707/RUP_RATIONNAL_UNIFIED_PROCESS

[28] http://sabricole.developpez.com/uml/tutoriel/unifiedProcess.

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

[30] https://netbeans.org/ 2016