Introduction générale

La sécurité des systèmes d'information est devenue un domaine crucial pour la survie de l'entreprise ou de toutes autres organisations. Les attaques réussies ou pas ne se comptent plus par milliers par an mais par centaines de milliers par semaine ce qui cause des dommages considérables socialement et financièrement…

Plusieurs travaux ont été élaborés afin de remédier à ce problème, ces travaux peuvent être classés en deux grandes catégories à savoir la sécurité fonctionnelle et non fonctionnelle.

Dans la première catégorie les chercheurs développent et améliorent des mécanismes et des outils afin de réduire les accès et empêcher les attaques tels que les firewalls, antivirus, etc.

Dans la deuxième catégorie on cherche à insérer les mesures de sécurité dans le cycle de vie depuis la collecte des besoins jusqu'au produit final en passant par la modélisation l'implémentation et les tests du logiciel afin d'assurer que les besoins de sécurité seront respectés.

Le contrôle d'accès est devenu un des mécanismes les plus imposants dans le domaine de la sécurité. Il restreint l'accès aux ressources du système suivant des contraintes préétablies (prédéfinies), en assurant plusieurs buts de la sécurité (confidentialité …), de ce fait plusieurs modèles ont été proposés tel que contrôle d'accès obligatoire (mandatoryaccessconrol MAC) …etc.

D'où notre idée d'intégrer la détection des failles liée aux accès non autorisés des systèmes en développement, et cela dès la phase de modélisation et plus particulièrement les systèmes modélisés en Unified Modeling language (UML).

Notre mémoire est divisé en quatre chapitres:

Chapitre 1:nousprésentons certaine notions de base de la sécurité des systèmes d'informations. Ensuite des rappels des différents concepts et des mécanismes de sécurité, et enfin la définitiondu modèle MAC.

Chapitre 2: nous introduisons le langage de modélisation UML, et les différents diagrammes ainsi que les méta-modèles et les mécanismes d'extension d'UML.

Chapitre 3 : nous commençons par la définition de notre application et les argumentations des choix faites sur le projet, ensuite nous passons à la partie modélisation qui inclut l'analyse et la conception de notre application, nous intégrons deux exemples pour dérouler les algorithmes utilisés.

Chapitre 4 : nous implémentons et testons notre application.

Conclusion : on fait la conclusion à partir de différents chapitre qu'on à vue précédemment.

Chapitre 1 : Sécurité et contrôle d'accès

Introduction

Le système d'information (SI) et les données sensibles d'une organisation, représentent son capital essentiel, qu'il convient de protéger contre les accès non autorisés. Les spécialistes desécurité des systèmes d'information (SSI) sont confrontés à des enjeux majeurs de sécurité. Ce chapitre nous permet de donner les concepts de base pour la compréhension de la sécurité et la définition de l'un des mécanismes les plus importants du contrôle d'accès qui est le contrôle d'accès obligatoire (Mandatory-Acces control-MAC).

I. Sécurité des systèmes d'information

I.1. Définition de sécurité des systèmes d'information

La sécurité des systèmes d'information est l'ensemble des mesures adoptées pour empêcher l'utilisation non autorisée, le mauvais usage, la modification ou le refus d'utilisation d'un ensemble de connaissances, de faits, de données ou de moyens. Le terme sécurité de l'information désigne donc les mesures préventives que nous mettons en place pour préserver nos informations et nos moyens. **[1]**

Le concept de sécurité des systèmes d'information recouvre un ensemble de méthodes, techniques et outils chargés de protéger les ressources d'un système d'information afin d'assurer:**[2]**

- **L'intégrité**: les services et les informations (fichiers, messages…) ne peuvent être modifiées que par les personnes autorisées (administrateurs, propriétaires…).
- **La confidentialité**: consiste à assurer que seules les personnes autorisées aient accès aux ressources échangées.
- **La disponibilité**: permet de maintenir le bon fonctionnement du système d'information et être disponible à chaque instant.
- **Le non répudiation:** permet de garantir qu'une transaction ne peut être niée.
- **L'authentification**: consistant à assurer que seules les personnes autorisées aient accès aux ressources.
- **La responsabilité**: regroupe la disponibilité et l'intégrité de l'identité de la personne qui a effectué l'opération.

I.2. Les mécanismes de sécurité

La sécurité des échanges passe notamment par un ensemble de techniques bien identifiées: **[2]**

- **La cryptographie:** les fonctions de base de la cryptographie sont le chiffrement et le déchiffrement. Lechiffrement vise à assurer la **confidentialité**d'informations.
- **Le hachage:** est un moyen de vérifier **l'intégrité** de l'information et donc prouver que les données transmises n'ont pas été modifiées entre la source et la destination.
- **Les contrôles d'accès:** permettent **l'authentification** des utilisateurs. Ils peuvent se faire par un mot de passe, une carte à puce, une clé, ou encore un élément biométrique.
- **La signature électronique:** permet non seulement d'assurer **l'authentification** de l'expéditeur et de vérifier **L'intégrité** du message reçu mais également de garantir la

non-répudiation (c'est-à-dire qu'elle permet d'assurer que l'expéditeur a bien envoyé le message).

I.3. Politique de sécurité

Une politique de sécurité est « l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique». Le développement d'une politique de sécurité peut être réalisé dans trois directions distinctes, à savoir : physique, administrative et logique. **[2]**

- **Physique** : cette politique précise un ensemble de procédures et de moyens qui protègent les locaux et les biens contre des risques majeurs (incendie, inondation, etc.) et contrôlent les accès physiques aux matériels informatiques et de communication (gardiens, codes, badges, ...).
- **Administrative** : cette politique définit un ensemble de procédures et moyens qui traite tout ce qui ressort de la sécurité d'un point de vue organisationnel au sein de l'entreprise (répartition des tâches, séparation des pouvoirs).
- **Logique :** c'est la politique de sécurité qui fait référence à la gestion du contrôle d'accès logique, lequel repose sur un triple service.

a. Service identification : identifier de façon unique un sujet grâce à un identifiant.

b. Service d'authentification : s'assurer que l'identité du sujet est bien celle qu'il prétend être.

c. Service d'autorisation : déterminer si le sujet authentifié peut effectuer l'action désirée sur l'objet spécifié. Le contrôle d'accès spécifie qui peut accéder à quoi et dans quelle circonstance, l'autorisation consiste à administrer et à examiner les droits d'accès, en fonction des spécifications de la politique de sécurité, ces spécifications sont généralement des :

- ✓ **Permission :** sujet a le droit de lire Objet.
- ✓ **Interdiction :** sujet n'a pas le droit d'écrire dans Objet.
- ✓ **Obligation :** sujet doit conserver Objet.

II. <u>contrôle d'accès</u>

II.1 Définitioncontrôled'accès

Un système de contrôle d'accès (figure 1) est un mécanisme qui autorise ou interdit à un sujet (entité capable d'initier des requêtes : personne, logiciel, etc.) l'accès à des objets (entités à protéger : fichiers, dossiers, etc.). Les décisions concernant ces autorisations et interdictions sont prises par un procédé de contrôle en conformité avec des règles de contrôle d'accès qui sont générer à partir d'une politique de sécurité. **[3]**

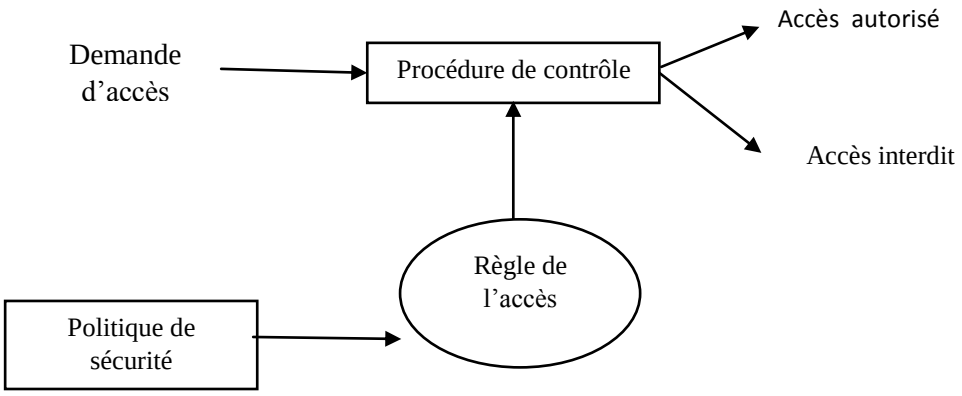


Figure 1: un système de contrôle d'accè**s**

II.2 Les éléments de base du contrôle d'accès

Les éléments de base du contrôle d'accès sont: les sujets (S) qui accèdent aux ressources (R) pour effectuer les actions (A).

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

II.3 Les politiques et les modèles de contrôle d'accès

Le modèle de contrôle d'accès est matérialisé par un ensemble de mécanismes. Il peut être défini comme un formalisme, souvent mathématique, qui consiste à vérifier si un sujet (utilisateur, processus …) demandant d'accéder à un objet (ressources...) possède les droits nécessaires pour le faire.

Le système de contrôle d'accès doit être capable de contrôler tout accès au système et à ses ressources en assurant les accès autorisés et seulement ceux-ci. Les schémas d'autorisation des politiques de sécurité sont classés en plusieurs modèles de contrôle d'accès, parmi eux le Mandatory-Acces control-MAC

II.3.1. Contrôled'accèsmandataire (MAC)

Le modèle MAC attribue l'accès des sujets aux objets à travers des niveaux de sécuritéattribué (figure 2).Il est utilisé principalement dans les environnements militaires à cause de son contrôle centralisé, il permet à l'administrateur du système de définir des privilèges pour protéger la confidentialité et l'intégrité des ressources dans le système. **[3]**
Les modèles associés au MAC sont:**modèle de Bell-LaPadula**, **modèle de Biba.[4]**

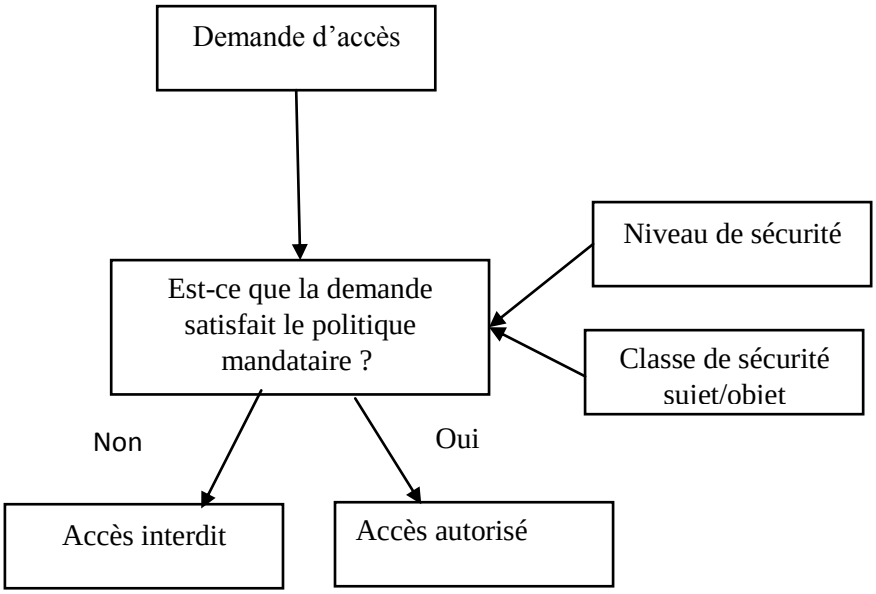


Figure 2: Le contrôle d'accès mandataire

II.3.1.1. Le modèle de Bell-LaPadula

Le modèle de Bell-LaPadulaa été développé par *David Bell* et *LeonardLaPadul*,L'objectif principal de ce modèleest de prévenir la divulgation d'informations au sein d'un système informatisé.Ce modèle assurer la confidentialité.Dans ce modèle les sujets et les objets sont regroupés par niveaux desécurité. Cela consiste àles classer en fonction de leurs importances dans lesystème. Par exemple les sujets et les objets peuvent être classés aux niveaux de sécurité suivants : **public, confidentiel, secret, top secret**.

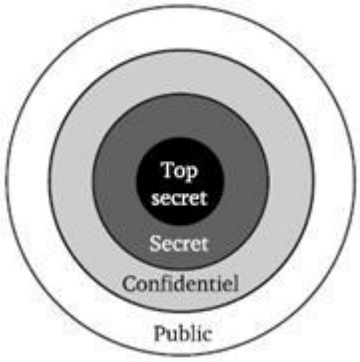


Figure 3: Les niveaux de sécurité dans MAC

L'importance des niveaux de sécurité est représentée par la relation d'ordre inférieure ou égale (≥). Nous pouvons par exemple ordonner les niveaux de sécurité précédents comme suite: **public ≤confidentiel ≤ secret ≤ top secret**.
Lorsqu'un sujet reçoit un niveau de sécurité, on parle de niveau d'habilitation.
La fonction d'habilitation est comme suite:f_s**: S** ⟶ **L**. S désigne l'ensemble des sujets et L l'ensemble des niveaux de sécurité.

De façon similaire lorsqu'un objet reçoit un niveau de sécurité, on parle de niveau de classification. La fonction de classification est :$\boldsymbol{f_o}$**: O** $\rightarrow$ **L**, avec O l'ensemble des objets.
Dans Bell LaPadula, les autorisations d'accès des sujets aux objets sont décidées sur la base de deux propriétés qui doivent être satisfaites. Il s'agit de la **propriété simple** et de la **propriété étoile.**

- **Propriété simple «*No read up*»:**un sujet *S* peut lire un objet O, si son niveau d'habilitation f_S est supérieur ou égal au niveau de classification$f_O$de l'objet**:**
  (*s, o,* lire) =>$\boldsymbol{f_S}$(s) $\geq$ $\boldsymbol{f_O}$(o).
- **Propriété étoile** ***«No write down»***: un sujet S peut écrire dans un objet O, si son niveau d'habilitation $f_S$est inférieur ou égal au niveau de classification$f_O$de l'objet :
  (*s, o,*écrire) =>$\boldsymbol{f_S}$(s) $\leq$ $\boldsymbol{f_O}$(o).

II.3.1.2. Politique de Biba

Ce modèle assure l'intégrité de l'information, ce nouveau modèle est conçu par Ken Biba, Dans ce modèle on parlera donc de niveau d'intégrité. Tout comme pour le modèle Bell LaPadulla, les accès des sujets aux objets dépendent respectivement de leurs habilitations et de leurs classifications. Pour garantir cela, les propriétés simples et étoile sont redéfinies.

- **Propriété d'intégrité simple «*No read down*»:** un sujet S peut lire un objet O, si l'habilitation de S est inférieure ou égale à la classification de O:
  (*s, o,*lire) =>$\boldsymbol{f_S}$(s) $\leq$ $\boldsymbol{f_O}$(o).
- **Propriété étoile *«No write up»*:** un sujet S peut écrire dans un objet O si l'habilitation de S est supérieure ou égale à la classification de O:**(*s, o,*écrire) =>$\boldsymbol{f_S}$(s) $\geq$ $\boldsymbol{f_O}$(o).**

III. Conclusion

Dans ce chapitre nous avons traité la sécurité des systèmes d'information en présentant le modèle de contrôle d'accès MAC. Ce modèle permet de présenter les règles de contrôle d'accès suivant les niveaux de sécurité. Dans le chapitre suivant nous présentons le langage de modélisation unifié UML, ses mécanismes d'extension ainsi que le méta modèles du diagramme de séquence.

Chapitre 2: UML (Unified Modeling Language)

Introduction

Après avoir introduit les concepts fondamentaux du contrôle d'accès, les mécanismes de sécurité, nous allons présenter, dans ce chapitre, le langage unifier de modélisation (UML) qui est le langage de modélisation orienté objet suivi de son historique et ses diagrammes, ensuite nous expliquons les méta-modèles pour mieux comprendre et étendre le modèle UML et nous finirons ce chapitre par la description des mécanismes d'extension et les profils UML.

I. Définition UML

UML (Unified Modelling Language) est un langage graphique permettant de représenter, spécifier, construire, et documenter les artefacts liés au développement d'un système à logiciel prépondérant.

UML propose une manière standard de représenter l'ensemble des éléments liés à la conception, depuis les aspects conceptuels avec les processus métier et les fonctions système, jusqu'aux éléments concrets tels que les structures de contrôle, les schémas de base de données, ou encore les composants logiciels réutilisables.**[5]**

II. Historique

Le langage de modélisation objet unifié UML, a été produit de la fusion des trois méthodes objet : **OMT** (***O**bject **M**odeling **T**echnique*) de **James Rumbaugh**, **OOSE** (***O**bject**O**riented **S**oftware **E**ngineering*) **d'Ivar Jacobson**, et la méthode de **GradyBooch**. En 1997, UML1.1 est devenu une norme **OMG** (***O**bject **M**anagement **G**roup*). La dernière version diffusée par l'OMG est UML 2.5 depuis mars 2015(voir Figure 4).**[5]**

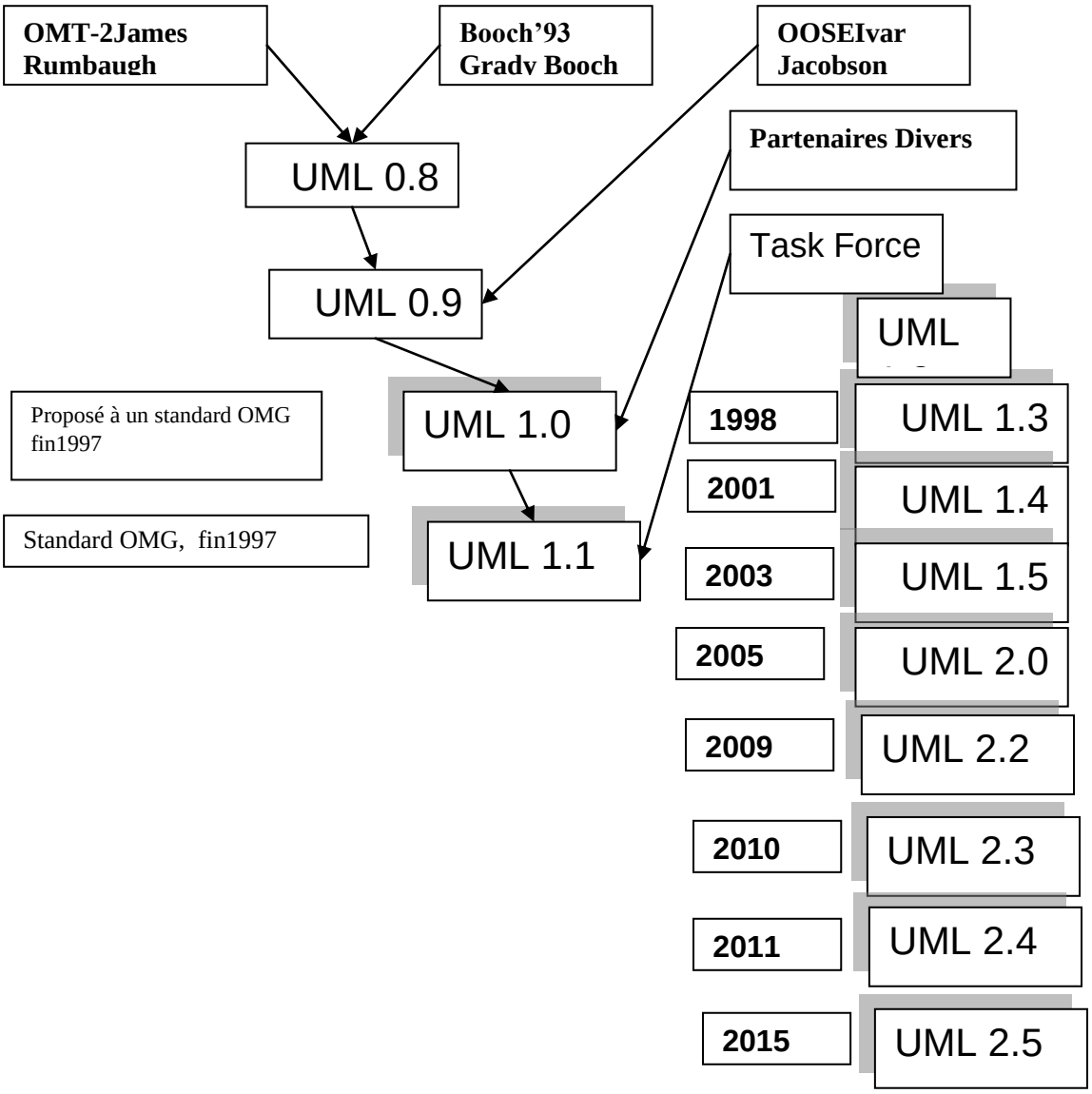


Figure4: Evolution des versions UML

III. Diagrammes UML

Langage UML propose 14 diagrammes dans sa version 2.5.

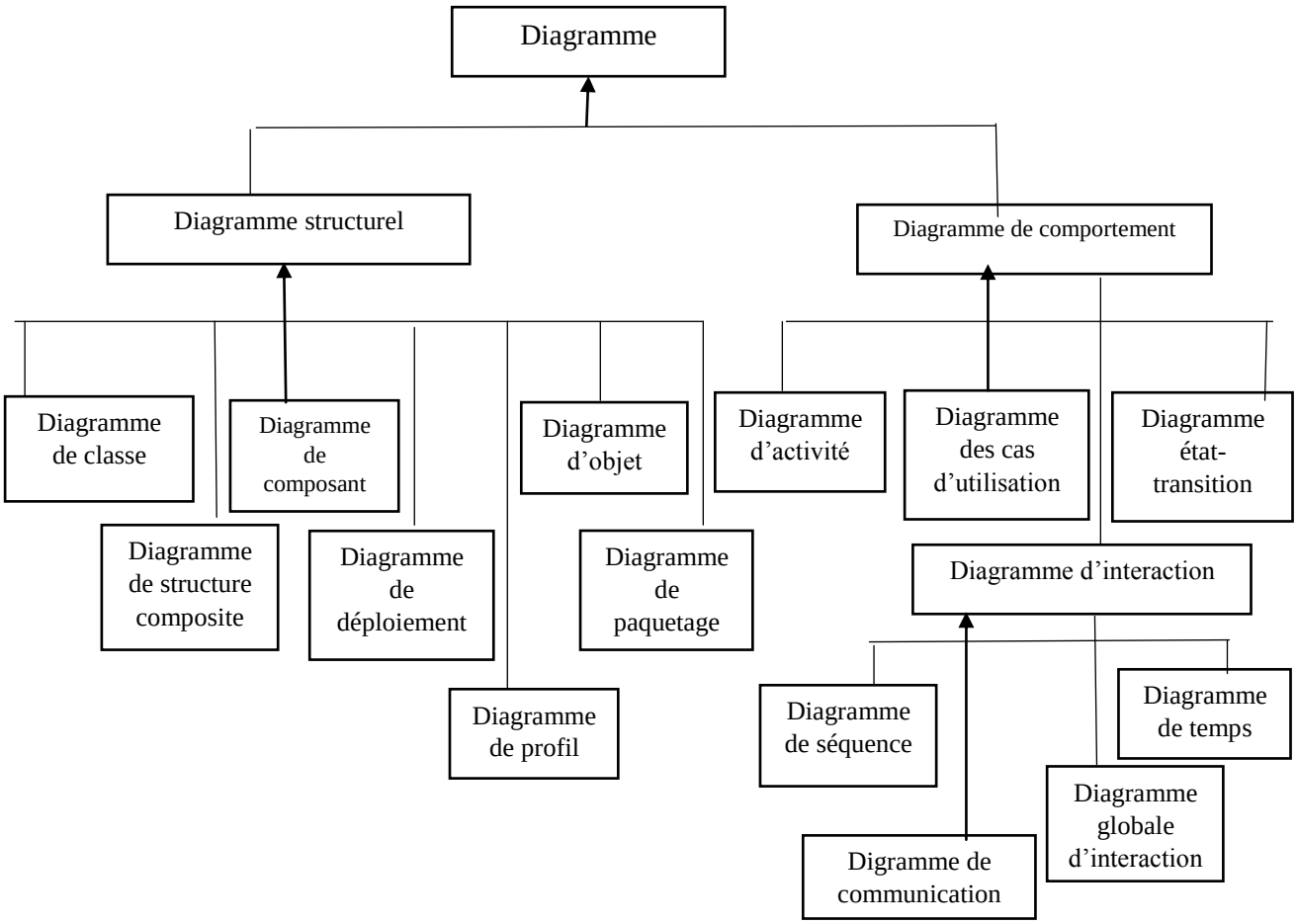


Figure 5 : La hiérarchie des diagrammes UML

La figure5, illustre les 14 diagrammes du la dernière version du langage UML. Certains de ces diagrammes permettent de représenter la structure d'un système (les aspects statiques); d'autres représentent son comportement (aspects dynamiques).**[9]**

Diagrammes structurels (statiques)

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

- ***Le diagramme d'objet :*** il sert à représenter les instances de classes (objets), Il montre des objets et les liens entre eux.

- ***Diagramme de composants :*** il montre les composants du système du point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques,..). Il permet de mettre en évidence les dépendances entre les composants (*qui utilisent quoi*).

- ***Diagramme de déploiement :*** il montre la disposition physique du matériel qui compose le système (ordinateurs, périphériques, réseaux...) et la répartition des composants sur ces matériels. Les ressources matérielles sont représentées sous forme de nœuds, connectés par des supports de communication.

- ***Diagramme des paquetages :*** un paquetage est un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML, il sert à représenter les dépendances entre paquetages.

- ***Diagramme de structure composite :*** est un ensemble d'éléments interconnectés collaborant dans un but commun lors de l'exécution d'une tâche, il est représenté par un ensemble de pièces (rôles) qui sont liées par des connecteurs.

- ***diagramme de profil*** : est une vue statique qui permet de décrire un mécanisme d'extension léger par rapport à UML pour répondre à un domaine particulier par exemple Java ou JEE ou Net.

Diagrammes comportementaux (dynamiques)

- ***Diagramme des cas d'utilisation :*** il permet de représenter les besoins des utilisateurs par rapport au système. Il montre les relations entre les acteurs et les cas d'utilisation du système.

- ***Diagramme d'activité :*** est une variante des diagrammes d'états-transitions. Il permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation. Dans un diagramme d'activité les états correspondent à l'exécution d'actions ou d'activités et les transitions sont automatiques.

- ***Diagramme états-transitions :*** permet de décrire sous forme de machine à états finis des comportements du système ou de ses composants. Il est composé d'un ensemble d'états, reliés par des arcs orientés qui décrivent les transitions.

- ***Diagramme de séquence :*** Il représente séquentiellement le déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs. Il sert à modéliser les aspects dynamiques des diagrammes des systèmes temps réels et des scénarios complexes. Dans ce type de diagramme, l'accent est mis sur la technologie des envois de messages.

- ***Diagramme de communication :*** c'est une représentation simplifiée d'un diagramme de séquence, en se concentrant sur les échanges de messages entre les objets.

- ***Diagramme global d'interaction :*** il fournit une vue générale des interactions décrites dans le diagramme de séquence et des flots de contrôle décrits dans le diagramme d'activités.

- ***Diagramme de temps :*** il permet de présenter l'interaction entre les objets actifs et leurs changements d'état sur un axe de temps. Il décrit les variations d'une donnée au cours du temps.

IV. Méta-modélisation

Depuis ses premières versions, le standard UML est caractérisé par sa sémantique définis par une approche de méta-modélisation. Un méta-modèle est la définition des constructions et des règles de création des modèles. Le méta-modèle d'UML définis donc la structure que doit respecter tout modèle UML.

Tout méta-modèle est spécifique à un domaine. Il est ainsi composé d'une terminologie et d'assertions. La terminologie est l'ensemble des concepts, propriétés et de leurs relations alors que les assertions sont des règles supplémentaires permettant de contraindre les éléments de la terminologie. Les concepts définis dans la terminologie doivent être suffisamment génériques pour être réutilisés dans plusieurs modèles. La possibilité de définis de multiples méta-modèles pour un même domaine a introduit le besoin d'un nouveau concept : le méta-méta-modèle. Son rôle est de fournir un langage unique pour la définition de tous les méta-modèles a fin de faciliter leur interopérabilité. Ce méta-méta-modèle est le MOF (Meta Object Facility) adopté en 1997 par l'OMG. L'approche de méta-modélisation adoptée par l'OMG est connue comme une hiérarchie à quatre niveaux.

- **Niveau méta-méta-modèle (M3):** est le niveau méta-méta-modèle, il définit le langage de spécification du méta-modèle. Le MOF (Meta Object Facility) est un exemple d'un méta-méta-modèle.
- **Niveau méta-modèle (M2):** est le niveau méta-modèle. Le méta-modèle d'UML se situe à ce niveau et il est spécifié en utilisant le MOF, c.à.d. les concepts du méta-modèle d'UML sont des instances des concepts de MOF. La figure 11 montre deux méta-classes du méta-modèle UML : Class et Association.
- **Niveau modèle (M1):** correspond au niveau des modèles UML des utilisateurs. Les concepts d'un modèle UML sont des instances des concepts du méta-modèle UML. La figure 6 montre un extrait de diagramme de classes pour une application bancaire contenant deux classes compte et client liées par une association UML. Les deux classes sont des instances de la méta-classe Class et le lien est une instance de la méta-classe Association du méta-modèle UML.
- **Niveau objets (M0):** correspond au niveau des objets à l'exécution. Il s'agit ici de deux objets cli et com des instances des deux classes Client et compte respectivement.

Le méta-modèle d'UML est décrit en utilisant une partie de la notation d'UML lui-même. Les concepts suivants sont utilisés: [**5]**

- ✓ 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.

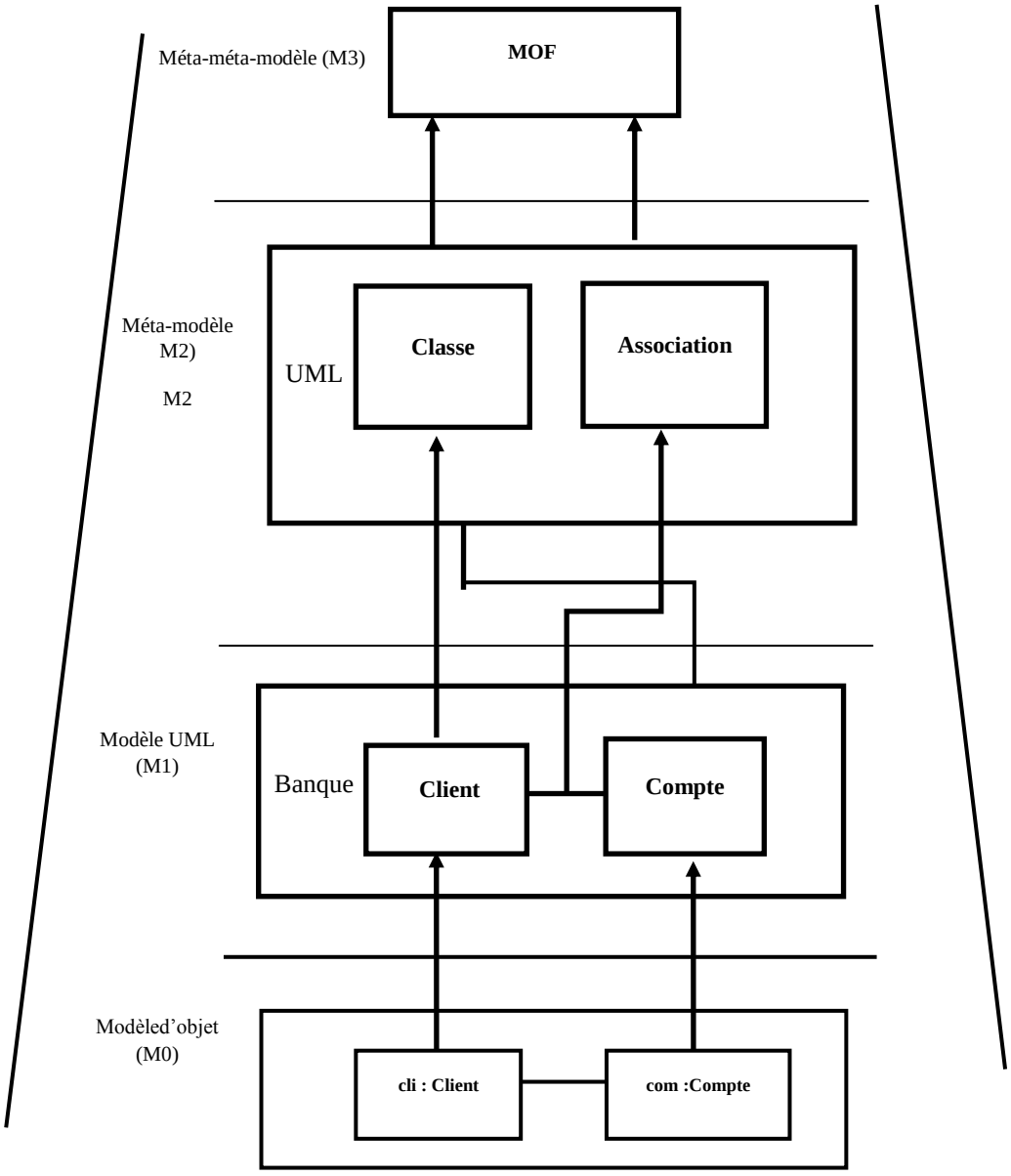


Figure 6: L'architecture à quatre niveaux de l'OMG

V. Méta-modèle de diagramme de séquence UML

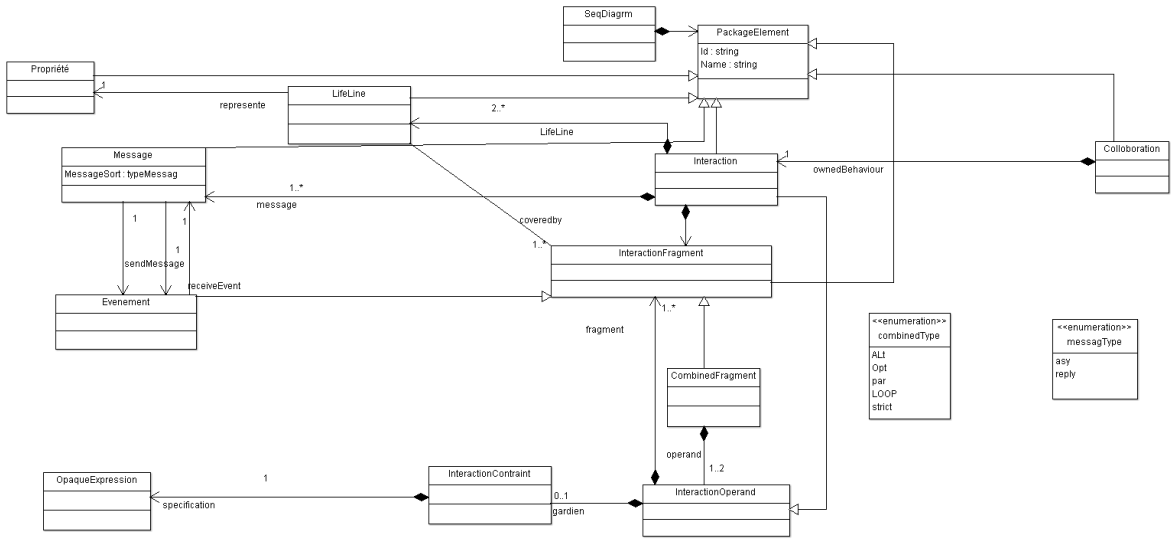


Figure 7: Méta-modèle de diagramme de séquence

La figure 7 montre le méta-modèle des diagrammes de séquence UML qui a été défini dans Ecore en utilisant l'éditeur EMF. Ce méta-modèle a été simplifié mais couvre la plupart des éléments importants. SeqDiagram représente un modèle de diagrammes de séquence UML, c'est la classe de haut niveau du méta-modèle. Le principal de l'élément graphique du diagramme est l'interaction**. [6]**

Lifeline, Message et InteractionFragment sont contenues dans Interaction. Lifeline, message et fragment sont les éléments de base des diagrammes de séquence UML. Lifeline représente un objet spécifique. Lifeline communiquent avec d'autres messages, chaque message déclenche deux événements: envoyer un événement et recevoir un événement. Un fragment est une instance de Event et CombinedFragment qui Héritent de la classe abstraite InteractionFragment. Les fragments décrivent les informations de comportement des diagrammes de séquence UML. Les événements sont les constructions comportementales de base des diagrammes de séquence UML et peuvent être combinés pour former des Constructions comportementales appelées CombinedFragment.
Un fragment combiné consiste d'un opérateur d'interaction, un ou plusieurs opérandes qui sont constitués d'événements Ou des fragments combinés, et une condition de garde optionnelle. Un fragment combiné Couvre un ensemble de Lifeline et décide du mode d'exécution et de l'état des fragments (Événements ou fragments combinés).

VI. Profile UML

La définition officielle d'un profil selon la dernière spécification d'UML (2.5) est la suivante **[5]**:
Un profil UML est un package dans lequel des éléments du méta-modèles (les méta-classes) vont être étendus. Le concept central du profil est le **stéréotype**, de **tagged value** et de **contraintes**. Il existe plusieurs profils standards EJB.

- **Stéréotypes:** un stéréotype se définit principalement sur les classes UML**.**
- **Tagged value:** les tagged value sont principalement utilisés pour ajouter des informations sur les classes.
- **Les contraintes**: sont utilisées pour exprimer des relations de stéréotypes et de tagged value.

VII. Lesmécanismesd'extensionsUML:

- **Les stéréotypes :** un stéréotype est utilisé pour créer des éléments de modélisation spécifique à un domaine. Le Nom du stéréotype est placé entre guillemets **(<<>>).**

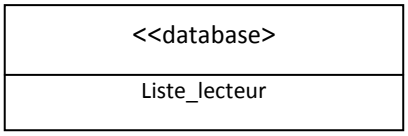


Figure 8: Présentation d'une classe stéréotypée

- **Les valeurs marquées (*tagged value)*:**une valeur marqué est un Couple (nom, valeur) associé à un élément du modèle, est principalement utilisé pour ajouter des informations sur les classes. Une valeur marquée est indiquée entre accolades.

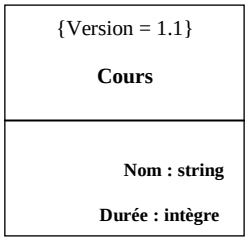


Figure 9 : Exemple d'une classe valeur marqué

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

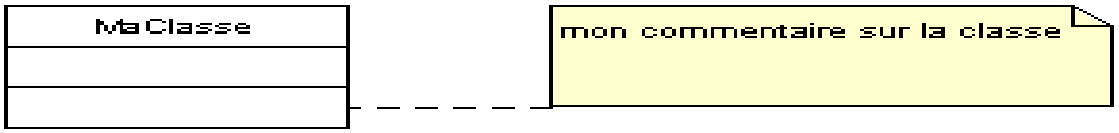


Figure 10: Présentation d'une note

- **Les contraintes** : une contrainteest une note ayant une valeur sémantique particulière pour un élément de la modélisation. Une contrainte s'écrit entre accolades {}.

Dans le cas où La contrainte concerne deux classes ou plus, celle-ci s'inscrit à l'intérieur d'une note. Dans UML, un langage spécifique d'expression de contraintes est disponible c'est le langage OCL (*Object Contraint Language*).

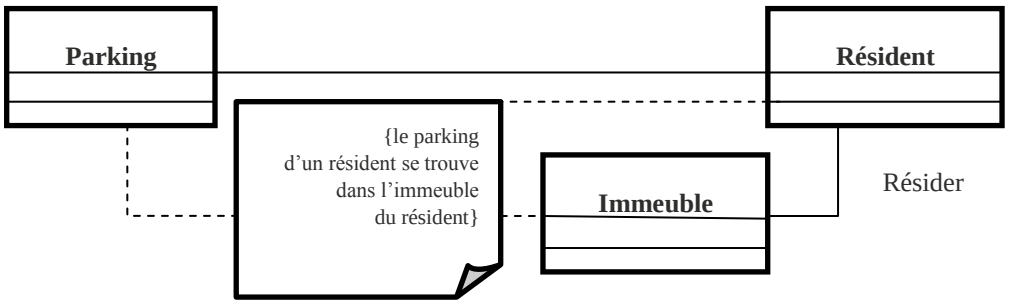


Figure 11: Exempled'utilisationd'unecontrainte

- **Profil:** afin de donner la possibilité de spécialiser chaque application d'UML à son propre contexte, UML propose de définir un **profil** d'utilisation caractérisé principalement par la liste des stéréotypes, la liste des valeurs marquées et les contraintes spécifiées pour un projet donné.

VIII. Conclusion

Ce chapitre nous a permis d'expliquerce qui est UML et ses diagrammes selon la version 2.5, nous nous sommes intéressés aux méta-modèles et UML profil pour comprendre comment étendre UML vers de nouveaux domaine, nous nous sommes intéressés plus particulièrement au méta-modèle du diagramme de séquence qui fait l'objet de notre études.

Chapitre 3: Analyse et conception

Introduction

Dans ce chapitre nous allons présenter la partie modélisation de notre application, pour cela nous allons passer par deux parties essentielles, dans la première partie nous allons définir l'application et argumenter les choix théoriques, dans la deuxième partie nous modélisant notre application en proposant la partie analyse et conception. Des algorithmes sont proposés pour résoudre le problème de détection des failles, et déroulés par deux exemples illustratifs.

I. Définition de l'application

Notre application permet de détecter les failles dans un système modélisé en UML (Diagramme de séquence).

Pour cela nous allons offrir à l'utilisateur une interface adéquate pour introduire (gérer) les éléments du modèle d'application (diagramme de séquence) et la politique de contrôle d'accès basée sur le MAC, le système détecte les failles du modèle d'application par la vérification de la cohérence entre le modèle d'application et la politique de sécurité introduite.

I.1. But de l'application

Cette application permet de détecter les failles d'un système dans la phase modélisation (avant la phase d'implémentation) son objectif est de détecter les failles selon la propriété de contrôle de d'accès MAC selon :

- Le modèle de Biba.
- Le modèle de LaPadula.

I.2. Cahier des charges fonctionnel

1-L'utilisateur introduit le modèle de son application :

- L'utilisateur saisit les objets et les interactions de diagramme de séquence.

2- L'utilisateur introduit la politique de sécurité de son application :

- L'utilisateur introduit la politique de contrôle d'accès selon le modèle MAC, cette politique sera décrite sur les éléments du diagramme de séquence.

3- Le système analyse la cohérence entre le modèle de l'application et la politique de contrôle d'accès introduite.

4-L'application affiche les accès non autorisés.

I.3. Pourquoi avons-nous choisi le diagramme de séquence ?

Notre choix c'est porté sur le diagramme de séquence pour les raisons suivantes: **[7] [8]**

- Le digramme de séquence définit des interactions entre objets à travers les échanges des messagesselon un ordre chronologique.

- Les principaux concepts de diagramme de séquence sont les objets (sujet et objet) et les messages (lecture ou écriture) qui sont la base de modèle de contrôle d'accès MAC.
- Les diagrammes de séquences sont très utilisés pour représenter le déroulement d'événements au fil du temps.
- Les diagrammes de séquence permettent de faire des tests sur l'application.
- Les diagrammes de séquence permettent de détecter les chemins infaisables. **[9]**

I.4. Pourquoi avons-nous choisi le modèle MAC ?

Nous avons choisi le modèle MAC pour les raisons suivante :

- Les éléments de modèle MAC sont clairs et bien définis.
- Le MAC utilise les deux propriétés pour la protection de l'intégrité et la confidentialité de l'information.
- Un haut niveau de sécurité, et donc une assurance élevée
- Si on partitionne les objets du modèle d'application (diagramme de sequence) comme sujets et objets du modèle MAC et les messages du modèle d'application comme des fonctions de lecture et écriture du modèle MAC, alors le MAC devient très approprié pour la définition de la politique de contrôle d'accès.

II. Vue statique de haut niveau de l'application

Notre projet est divisé structurellement en trois packages comme le montre la figure 12 ci-dessous

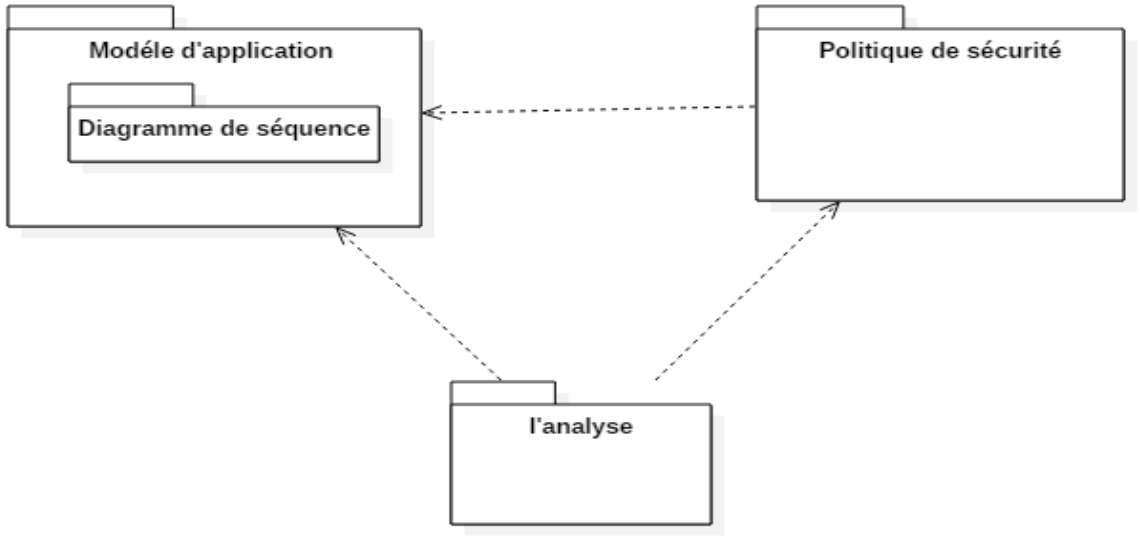


Figure 12 : Diagramme de package de haut niveau de l'application MAC.

- **Modèle d'application:** ce package regroupe les propriétés du diagramme de séquence.

- **Politique de sécurité:** ce package regroupe les données qui permettent d'exprimer les la politique de contrôle d'accès sur les éléments du modèle d'application (diagramme de séquence).

- **Analyse:** ce package regroupe essentiellement les deux propriétés de politique de sécurité MAC (modèle de padula et modèle de biba) pour permettre d'analyser et détecter les failles, les résultats de la vérification de la cohérence dépend du modèle d'application et de la politique de sécurité.

III. Vue dynamique de haut niveau de l'application

Afin de bien comprendre notre application nous proposons une vue dynamique de haut niveau, cette vue nous allons l'illustrer par quatre diagrammes d'activités :

- Diagramme d'activité de haut niveau de l'application.
- Diagramme d'activitéde haut niveau gérer modèle d'application de l'application.
- Diagrammed'activité de haut niveau gérer politique de sécurité.
- Diagramme d'activité de haut niveau d'analyse.

III.1. Diagramme d'activité de haut niveau de l'application

La figure 13 montre les activités faite par l'utilisateur pour dérouler notre application. L'utilisateur gèrent les éléments de modèle d'application (les éléments du diagramme sequence) puis gèrent les éléments de modèle de contrôle d'accès MAC et en fin il lance l'analyse, les résultats sont affichés.

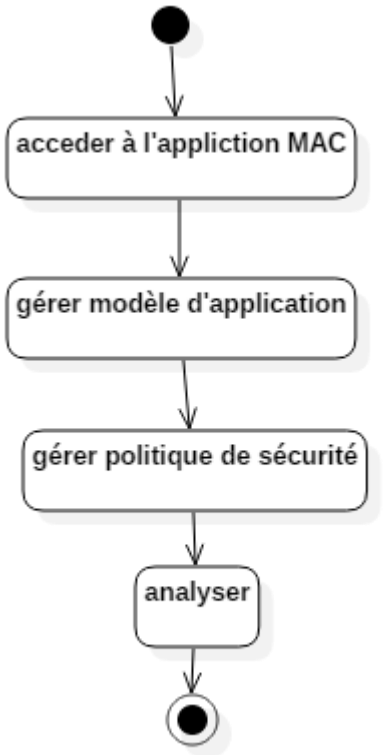


Figure 13: Diagramme d'activité de haut niveau de l'application**.**

III.2. Diagramme d’activité de haut niveau gérer modèle d'application

La figure 14montre une succession d’étapes pourgérer le modèle d’application cette gestion impose à l’utilisateur de faire la gestion du diagramme de séquence.

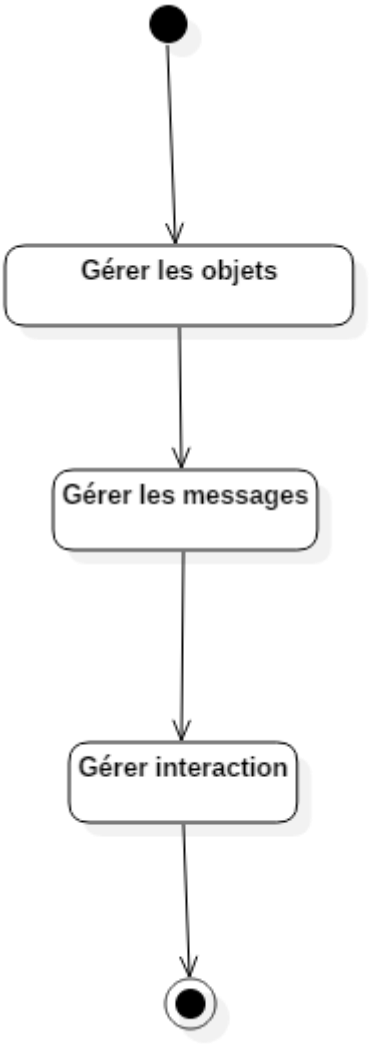


Figure 14: Diagramme d’activité de haut niveau gérer modèle d'application de l'application.

III.3. Diagramme d’activité de haut niveau gérer la politique de sécurité

La figure 15montre une succession d’étapes pour gérer le politique de sécurité, cette gestion impose à l’utilisateur de commencer par la gestion des niveaux de sécurités et le partitionnement des sujets et objets et donner les niveaux de sécurités et à la fin on gère les types des messages.

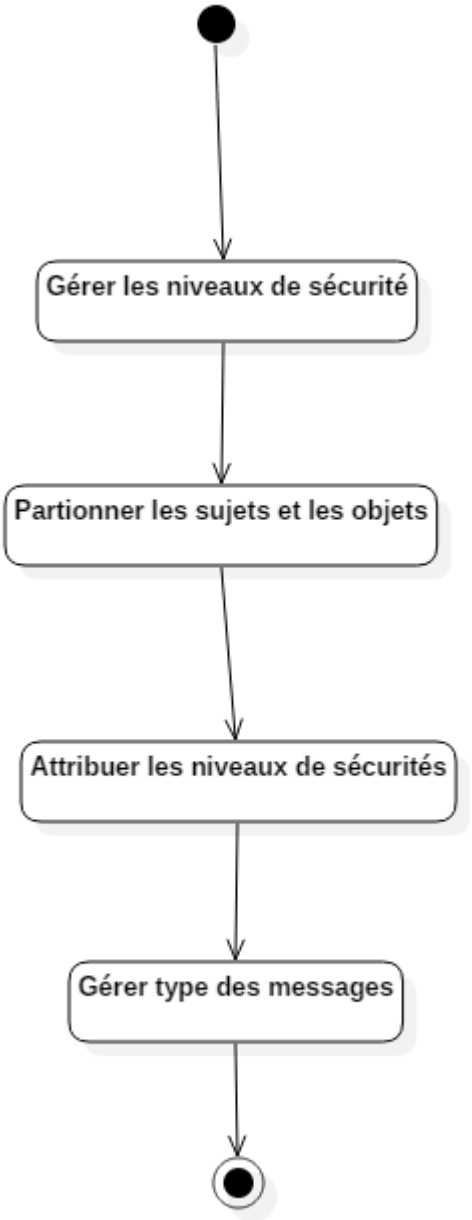


Figure 15: Diagramme d'activité de haut niveau gérer politique de sécurité.

III.4. Diagramme d'activité de haut niveau d'analyse

Ce diagramme figure 16montres une succession d'étapes pour analyser, détecter et afficher les problèmes de sécurité. Ces étapes montrent les algorithmesqui permettent de déceler les failles du système par rapport les deux modèles de MAC.

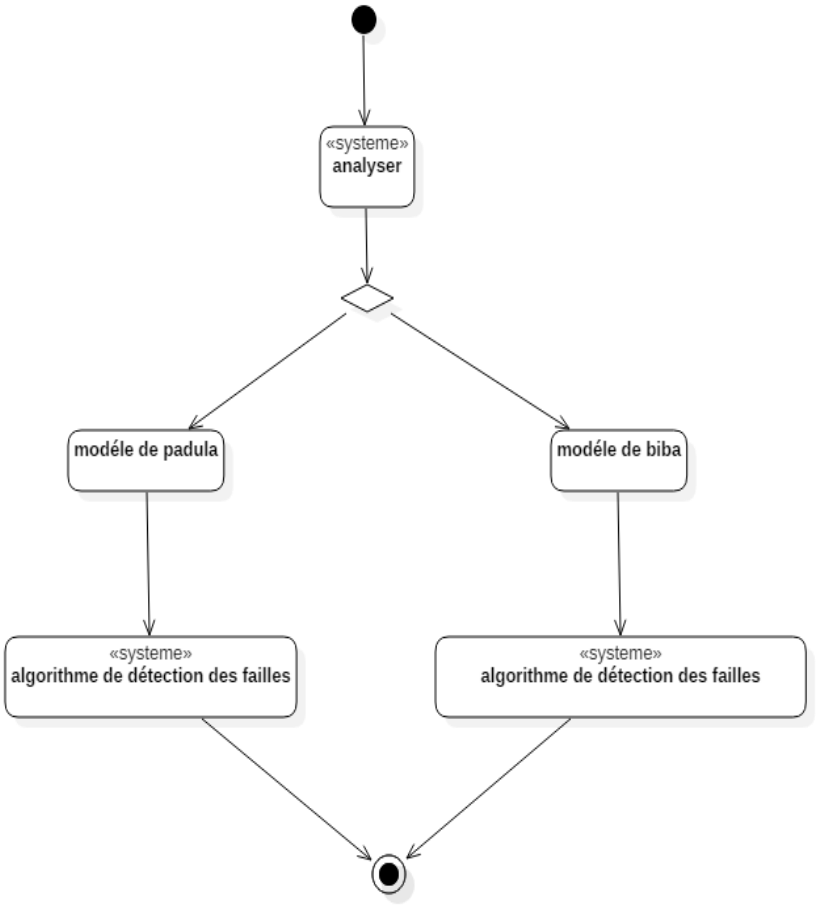


Figure 16: Diagramme d'activité de haut niveau d'analyse.

IV. 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 UnifiedProcess).

IV.1. RUP

Le RUP (rational unifiedprocess) est un processus d'ingénierie logicielle. Il fournit une approche disciplinée pour assigner des taches et des responsabilités au sein d'une organisation de développement. Son objectif est d'assurer la production de logiciels de haute qualité qui répondeau besoin de ses utilisateurs.**[10]**

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

Notre modélisation passe par deux parties importantes analyse et conception

Notre modélisation passe par deux parties importantes analyse et conception.

1- **L'analyse des besoins** : qui se divise en quatre étapes :
- La collecte des besoins.
- La spécification des cas d'utilisation.
- Description textuelle de chaque cas d'utilisation.
- Diagramme de classe d'analyse.

2-La conception: elle se divise en quatre étapes :
- La spécification de diagramme de classe de conception.
- La spécification de diagramme de séquence de conception.
- La spécification de diagramme d'activité

3- **L'implémentation** : Cette phase sera décrite dans le chapitre suivant.

IV.2. Analyse des besoins

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

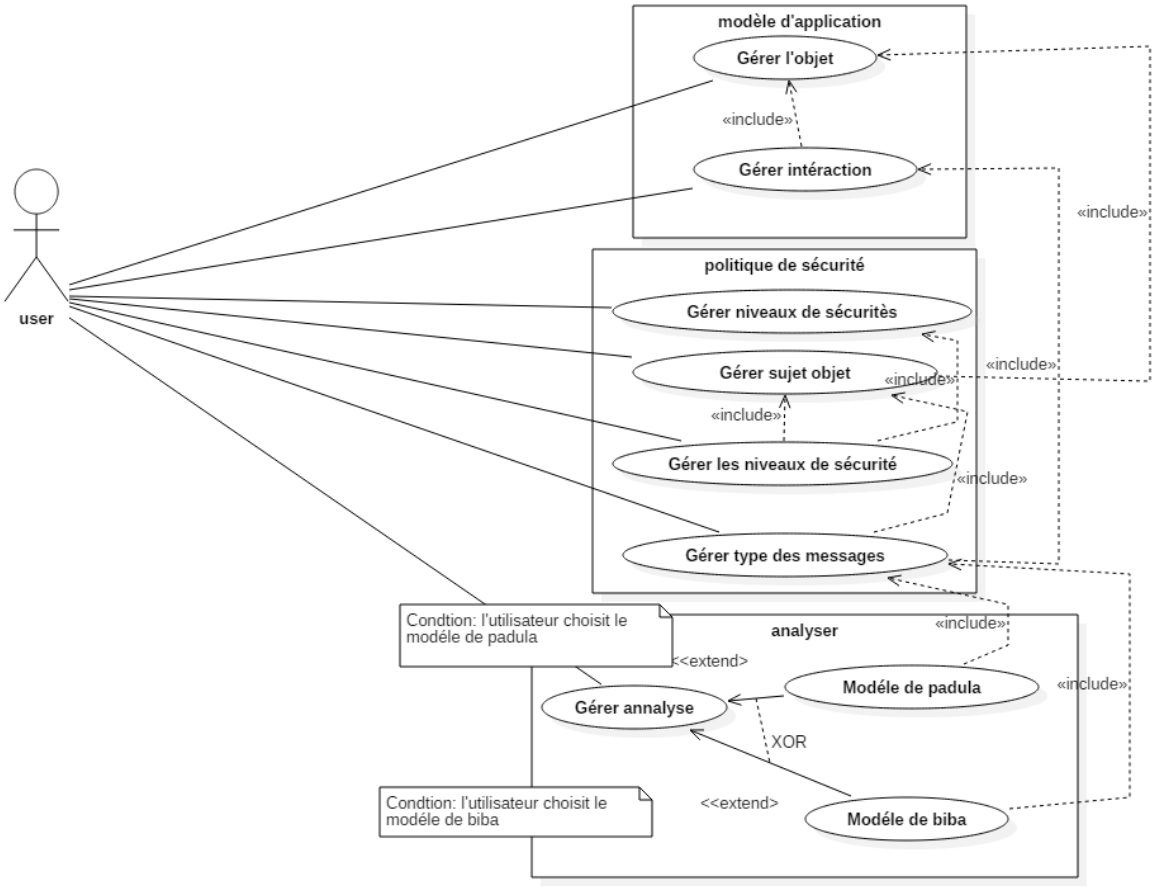


Figure 17: Diagramme de cas d'utilisation de notre application.

IV.2.2.Description textuelle de diagramme de cas d'utilisation de l'application

Description textuelle : gérer diagramme de séquence (modèle d'application)

Titre : Gérer les objets.

Acteur principale :

- Utilisateur

Objectifs :

- Ce cas d'utilisation permet à un utilisateur de rentrer les objets, de supprimer ou encore en modifier les objets ou supprimer tous les objets, si les objets existent.

  Précondition : Gérer les objets(matrice d'interaction vide).

Enchainement nominal :

- L'utilisateur accède à l'application.
- L'utilisateur choisit modèle d'application.
- L'utilisateur choisit diagramme de séquence.
- L'utilisateur choisit objet.
- L'utilisateur saisit les noms d'objets.
- L'utilisateur valide les éléments introduits.

Post condition : Objet est ajouté (afficher matrice d'interaction).

Enchainement alternative

Pré condition : Objet existe déjà.

A1-modifier un objet.

- L'utilisateur sélectionne un objet.
- L'utilisateur modifie.
- L'utilisateurdémarre de point 5 de séquence nominale.

Post condition : Objet modifié.

A2-Supprimer un objet.

- L'utilisateur sélectionne un objet.
- L'utilisateur supprime.

Post condition : Objet supprimé.

A3-Supprimer tous les objets.

- L'utilisateur clique sur supprimer tous.

Post condition : Tous les objets sont supprimés.

Enchainement d'exception

E1-verifier vide

- L'enchainement démmare à partir de point 1 de séquence nominale
- Le systéme indique Le champ est vide.

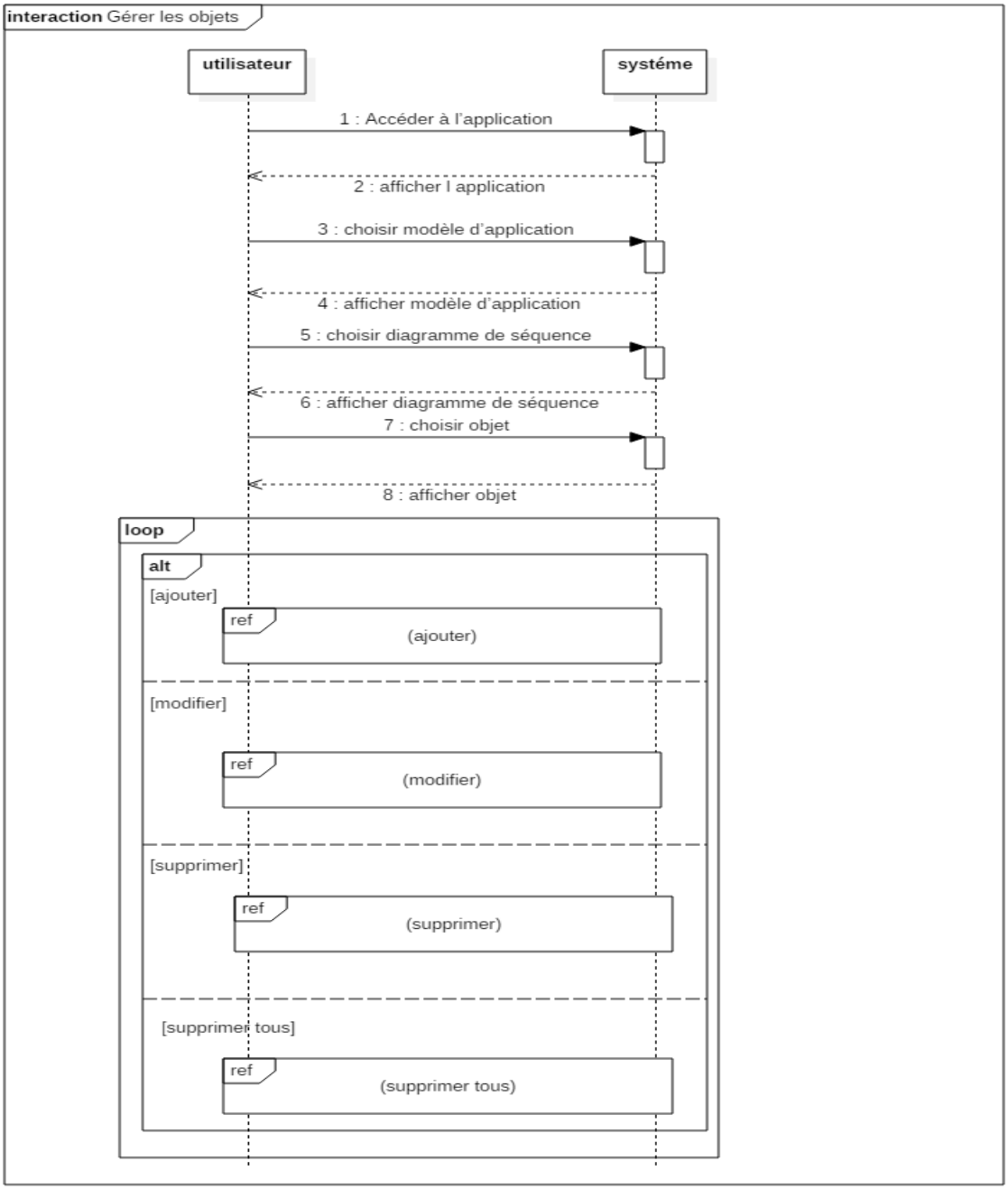


Figure 18: diagramme de séquence de haute niveau pour gérerles objets

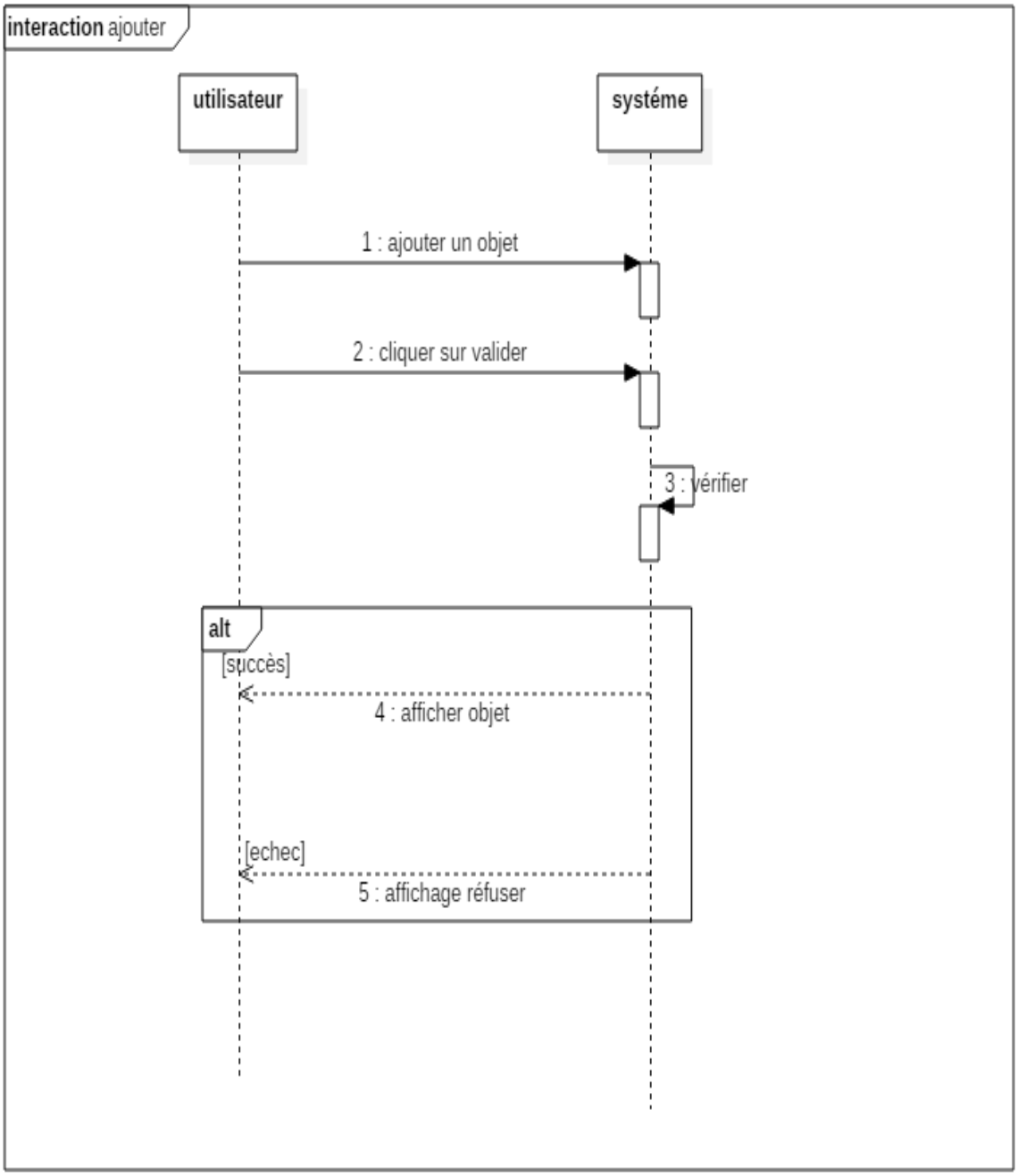


Figure 19: diagramme de séquence de haute niveau pour gérer les objets (ajouter)

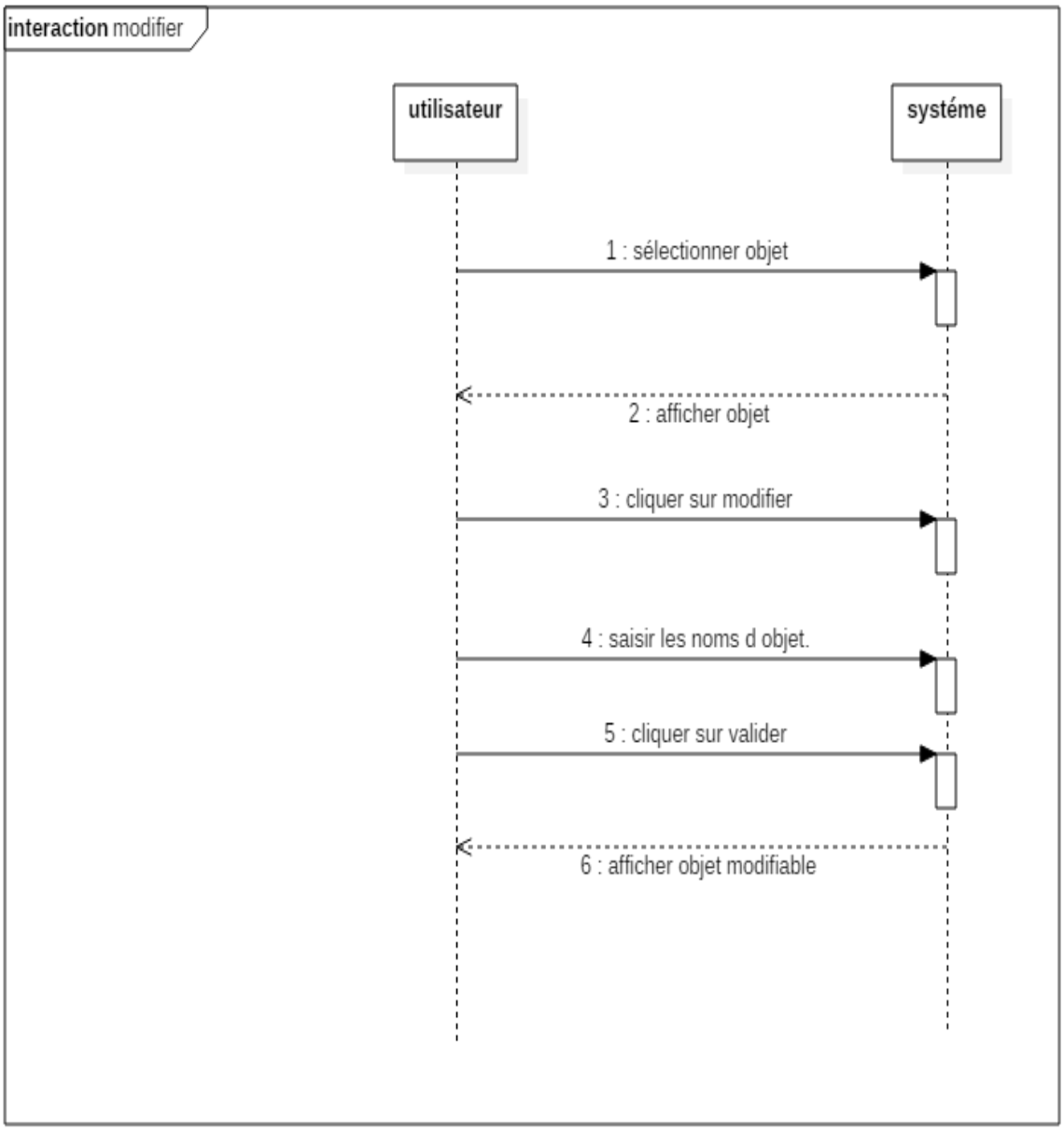


Figure 20: diagramme de séquence de haute niveau pour gérer d'objet (modifier)

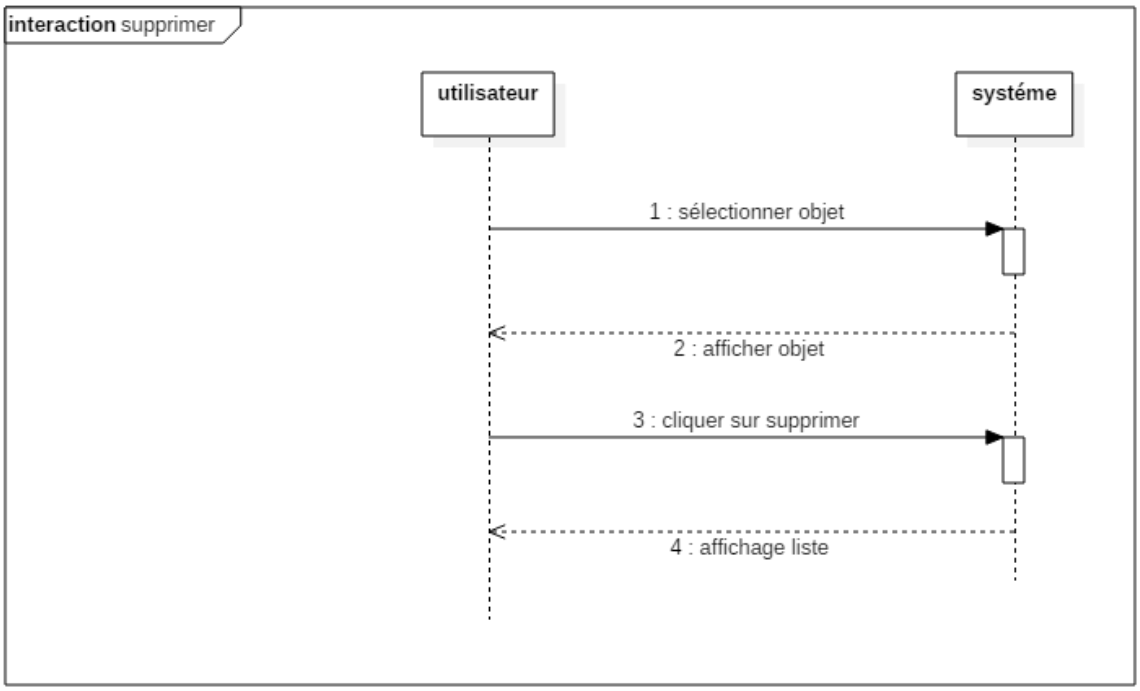


Figure 21: diagramme de séquence de haute niveau pour gérer d'objet (supprimer)

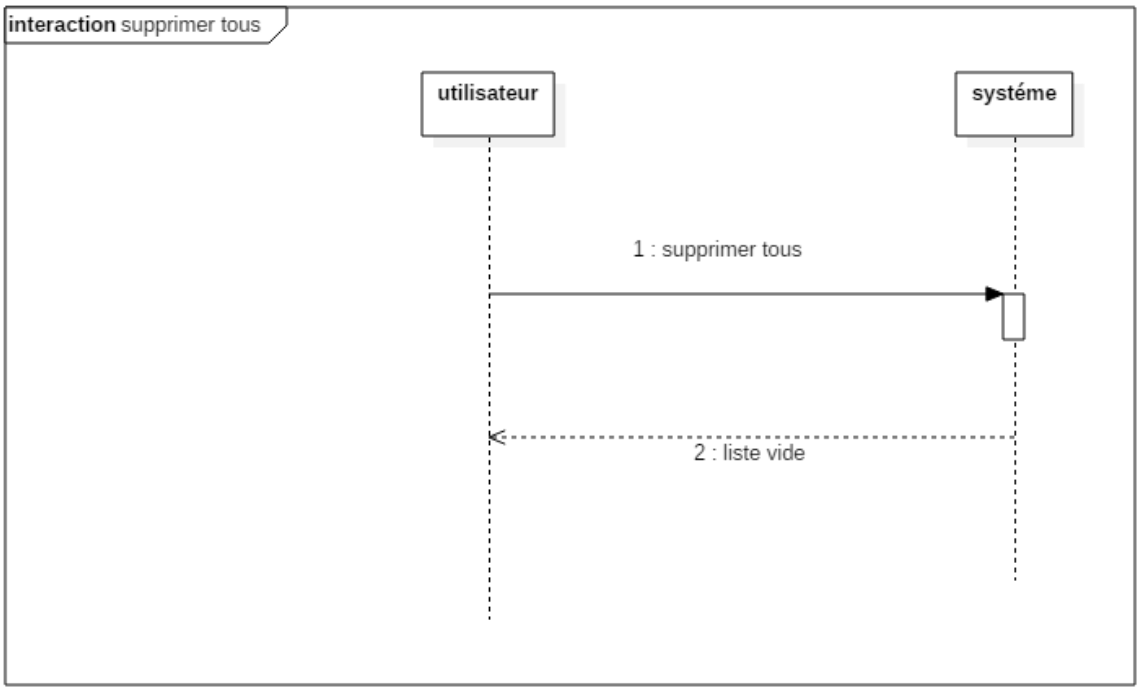


Figure 22: diagramme de séquence de haute niveau pour gérer d'objet (supprimer tous)

Effectuer des messages entre les objets (interaction)

Titre : Gérer l'interaction.

Acteur principale :

- Utilisateur

Objectifs :

- Ce cas d'utilisation permet à un utilisateur de choisir les objets envoyer et les objets reçue, saisir les messages entre les objets.

Pré condition : Gérer l'interaction.

Enchainement nominal :

- L'utilisateuraccède à l'application.
- L'utilisateur choisitl'interaction.
- L'utilisateur choisitcombined fragment.
- L'utilisateur choisit les objets.
  -objet envoyé.
  -objet reçu.
- L'utilisateur saisit un message.
- L'utilisateur valide.
- L'utilisateur clique sur fin de combined fragment.

Post condition :l'interaction est ajoutée.

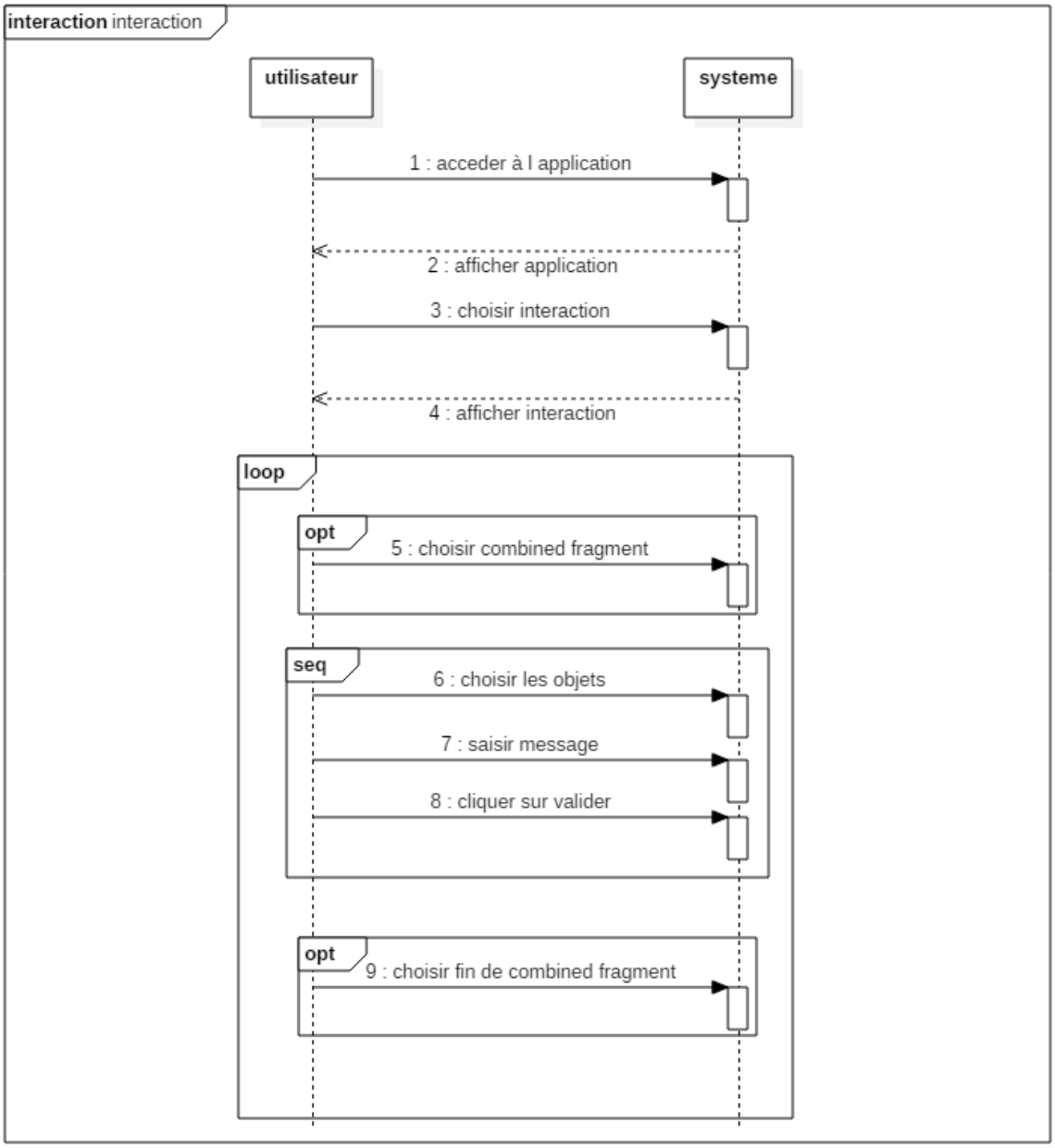


Figure 23: diagramme de séquence de haute niveau pour gérer l'interaction

Description textuelle : gérer politique de sécurité

Titre : Gérer les niveaux de sécurité**.**

Acteur principale :

- Utilisateur

Objectifs :

- Ce cas d'utilisation permet à un utilisateur de saisir les niveaux de sécurités, de supprimer ou encore en modifier les niveaux de sécurités, si les niveaux de sécurités existent.

Pré condition : Gérer les niveaux de sécurités.

Enchainement nominal :

- L’utilisateur accède à l’application.
- L’utilisateur choisit politique de sécurités.
- L’utilisateur choisit MAC.
- L’utilisateur choisitniveau de sécurité.
- L’utilisateur saisit les noms de niveau de sécurité.
- L’utilisateur valide.

Post condition :

- Niveau de sécurité est ajouté.

Enchainement alternative

Pré condition : Niveau de sécurité existe déjà.

A1-Modifier un niveau de sécurité

- L’utilisateur sélectionne un niveau de sécurité.
- L’utilisateur modifie.
- L’utilisateur démarre de point 5 de séquence nominale.

Post condition

- Niveau de sécurité modifié.

A2-Supprimer un niveau de sécurité.

- L’utilisateur sélectionne un niveau de sécurité.
- L’utilisateur supprime.

Post condition

- Niveau de sécurité supprimé.

A3-Supprimer tous les niveaux de sécurités.

- L’utilisateur supprimé tous.

Post condition

- Tous les niveaux de sécurités sont supprimés.

Enchainement d’exception

E1-verifier vide

- L’enchainement démmare à partir de point 1 de séquence nominale
- Le systéme indique que le champ est vide.

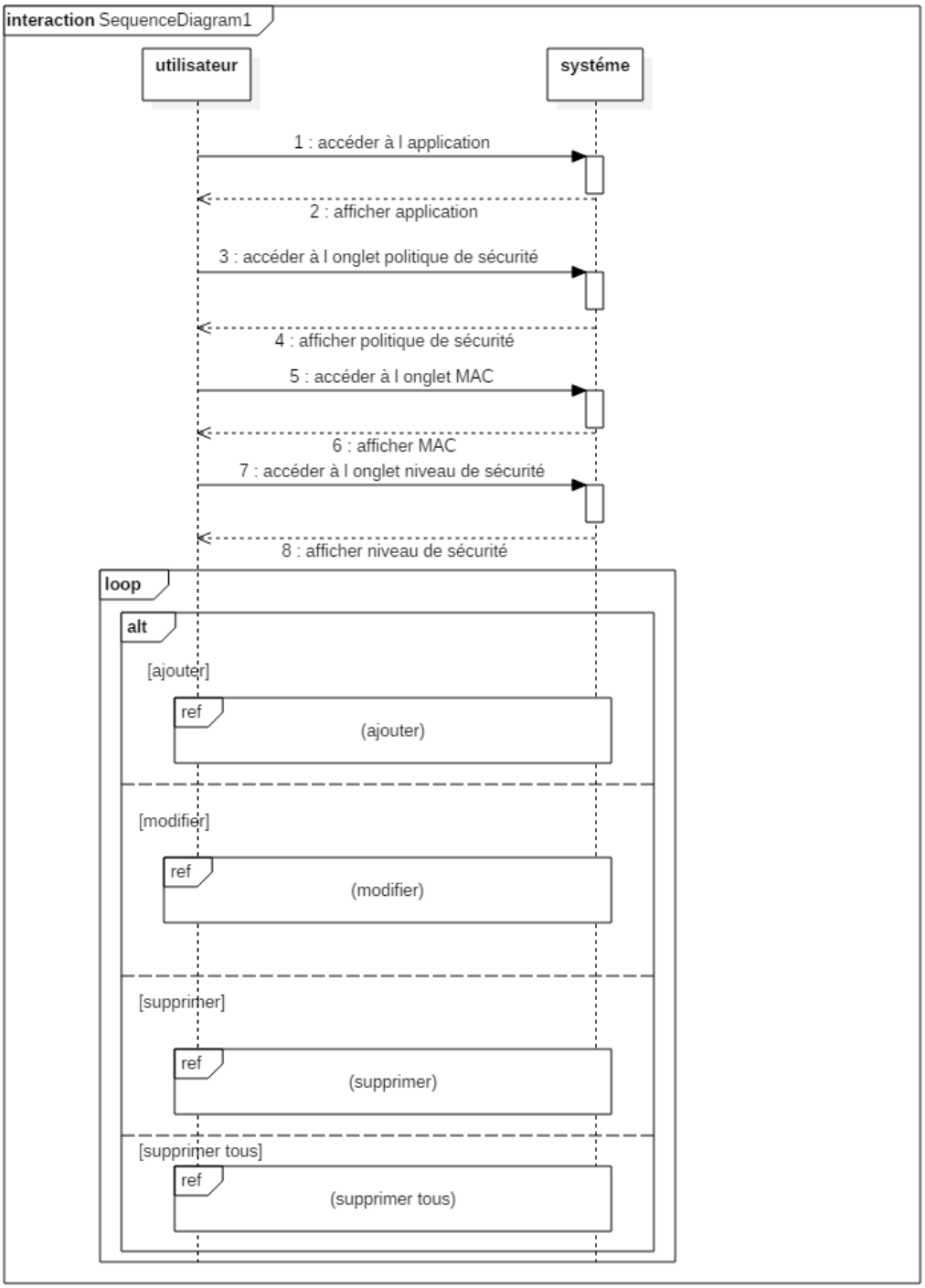


Figure 24: diagramme de séquence de haute niveau pour gérer les niveaux de sécurité

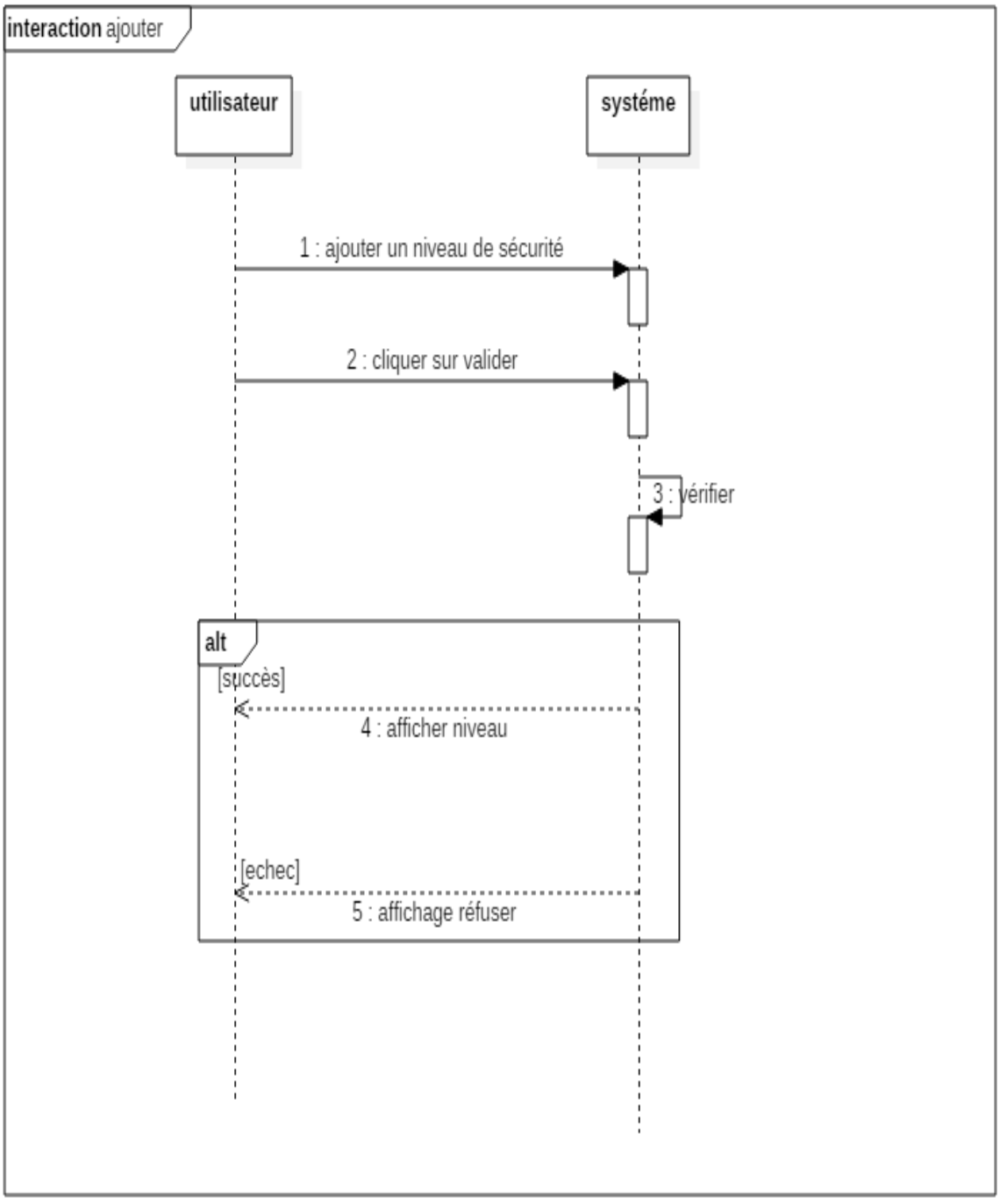


Figure 25: diagramme de séquence de haute niveau pour gérer les niveaux de sécurité(ajouter)

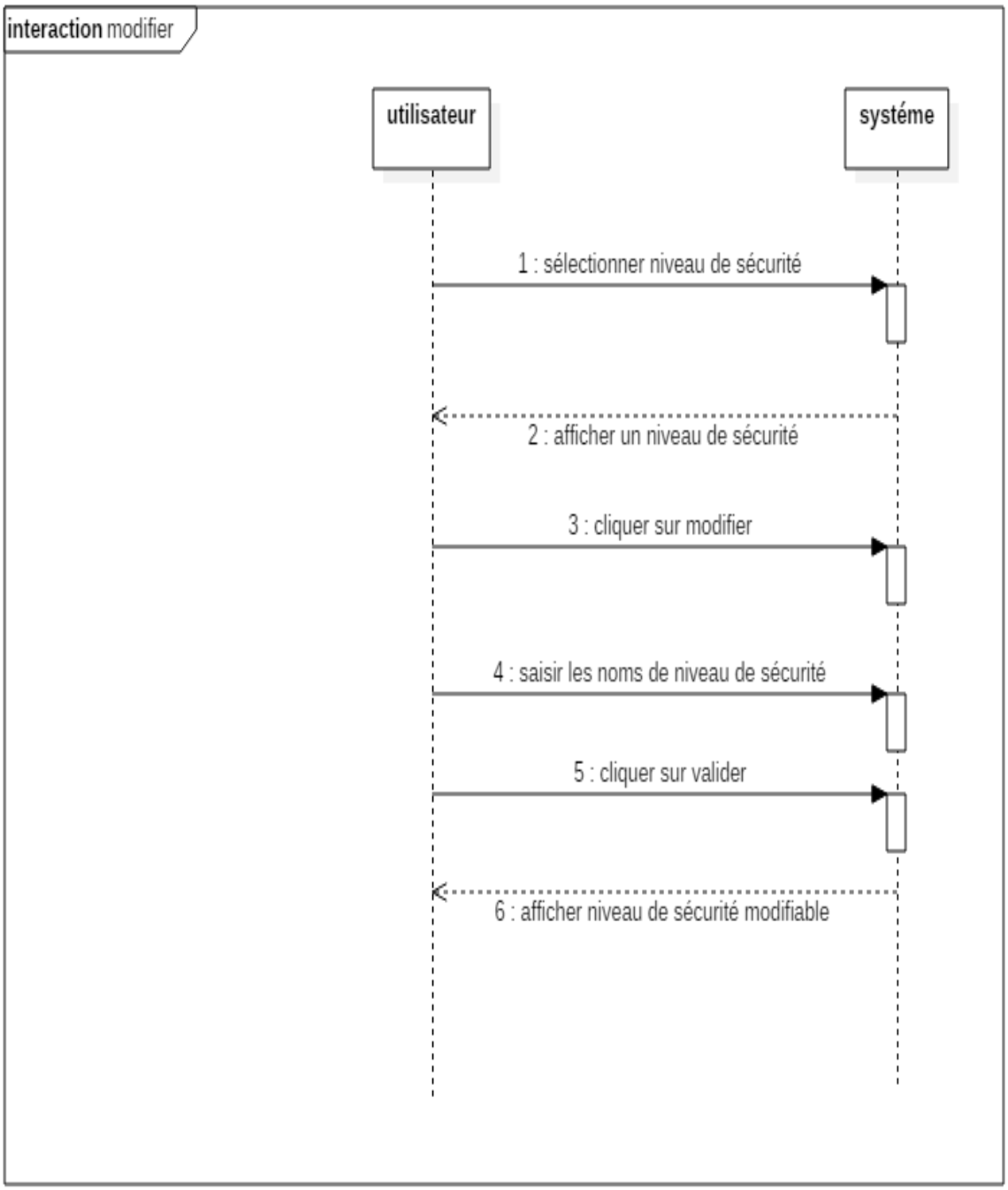


Figure 26: diagramme de séquence de haute niveau pour gérer les niveaux de sécurité(modifier)

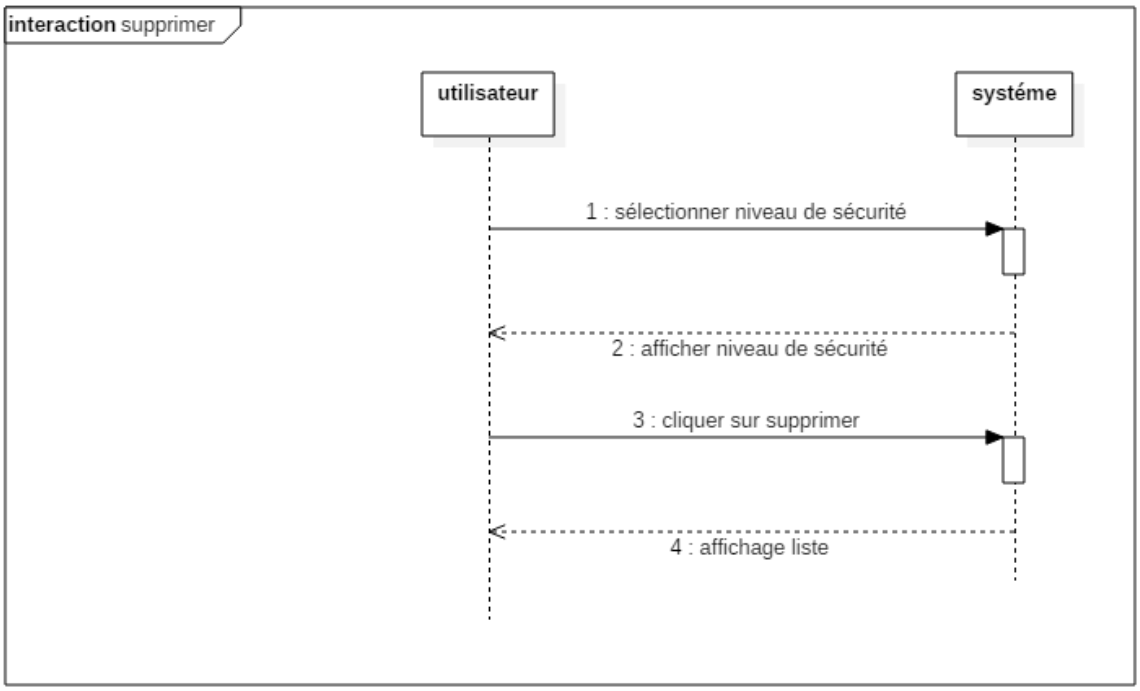


Figure 27: diagramme de séquence de haute niveau pour gérer les niveaux de sécurité(supprimer)

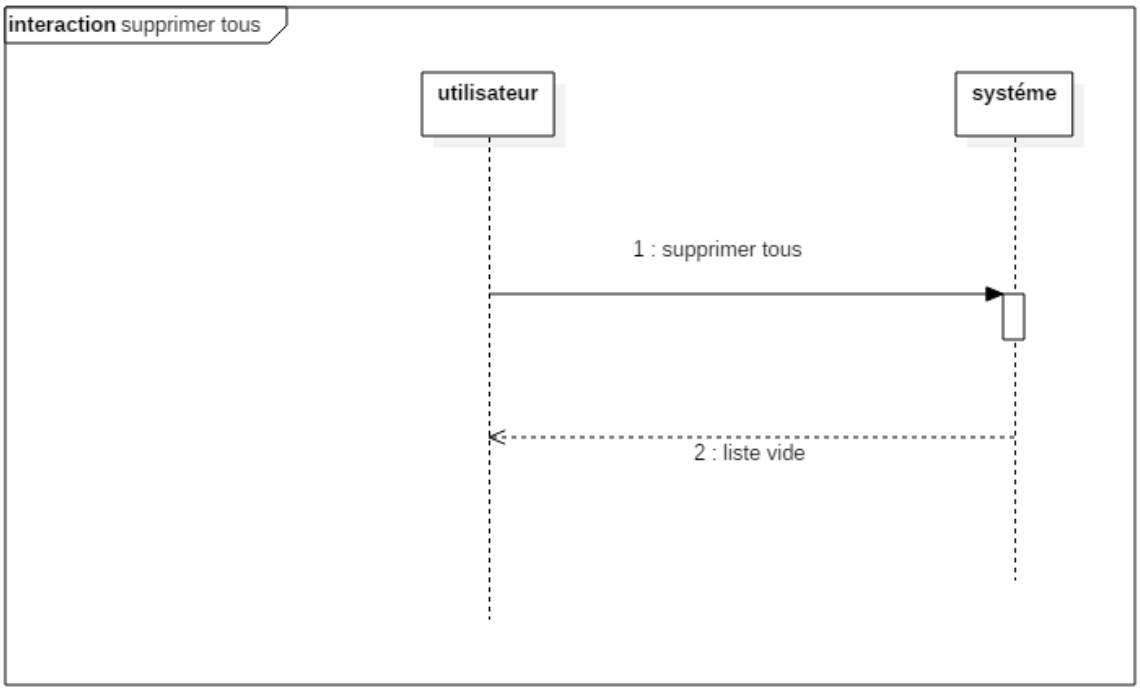


Figure 28: diagramme de séquence de haute niveau pour gérer les niveaux de sécurité(supprimer-tous)

Titre : Partitionner les sujets et les objets

Acteur principale :

- Utilisateur.

Objectifs :

- A tout moment l'utilisateur peut accéder à la liste d'objets, et partitionner les sujets et les objets.

Pré condition : Partitionner sujet et objet

Enchainement nominal :

- L'utilisateur accède à l'application.
- L'utilisateur choisit partitionner sujet objet.
- L'utilisateur sélectionne un objet.
- L'utilisateur sélectionne un type d'objet.
  -Sujet.
  -Objet.
- L'utilisateur valide.

Post condition : Le partitionnement est fait.

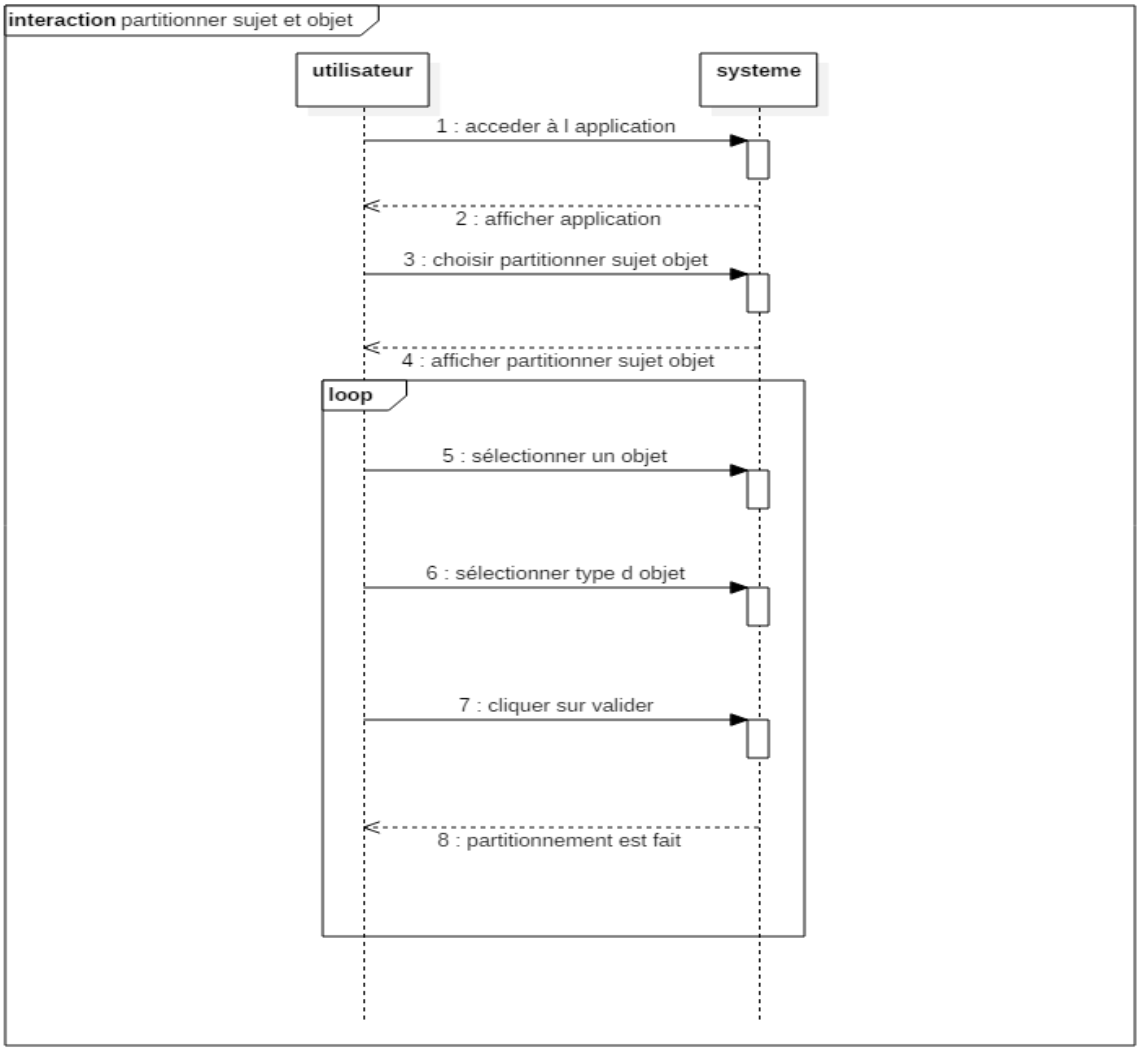


Figure 29: diagramme de séquence de haute niveau pour partitionner les sujets et les objets

Titre : Attribuer un niveau de sécurité pour les sujets et les objets

Acteur principale :

- Utilisateur.

Objectifs :

- A tout moment l'utilisateur peut accéder à la liste des objets et les sujets, et sélectionner un niveau de sécurité.

Pré condition : Niveau de sécurité pour sujet et objet.

Enchainement nominal :

- L'utilisateur accède à l'application.
- L'utilisateur choisit niveau de sécurité sujet objet.
- L'utilisateur sélectionne un sujet.
- L'utilisateur sélectionneun niveau de sécurité pour un sujet.
- L'utilisateur valide.
- L'utilisateur sélectionne un objet.
- L'utilisateur sélectionneun niveau de sécurité pour un objet.
- L'utilisateur valide.

Post condition : Les niveaux est donnés pour les sujets et les objets.

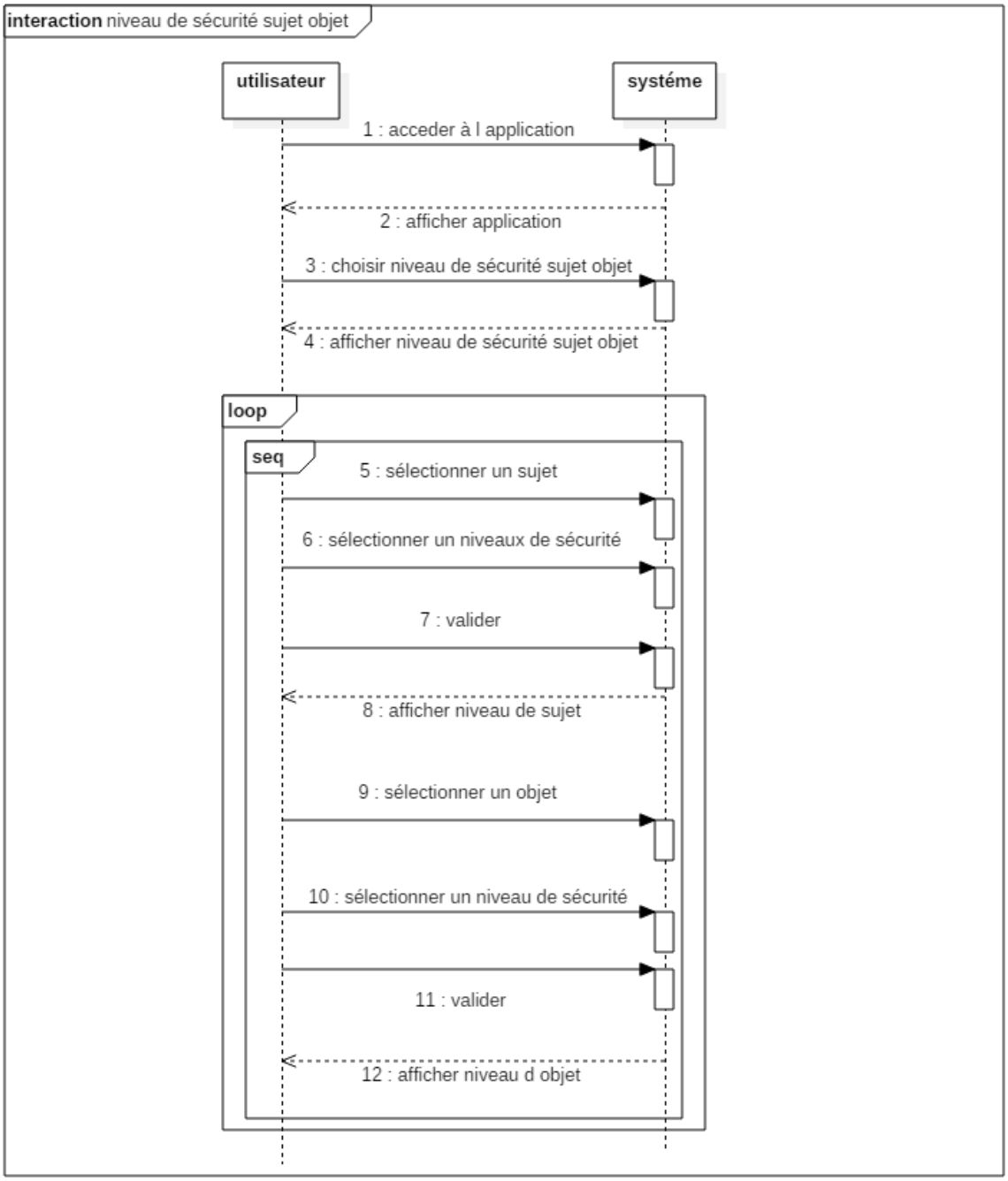


Figure 30: diagramme de séquence de haute niveau donner les niveaux pour les sujets et les objets.

Donner les types des messages

Acteur principale :

- Utilisateur.

Objectifs :

- A tout moment l'utilisateur peut donner les types des messages existe pour une interaction.

Pré condition : Type de message.

Enchainement nominal :

- L’utilisateur accède à l’application.
- L’utilisateur choisit type de message.
- L’utilisateur sélectionne uneinteraction.
- L’utilisateur choisit un type de message.
- L’utilisateurvalide.

Post condition : le type de message est donné

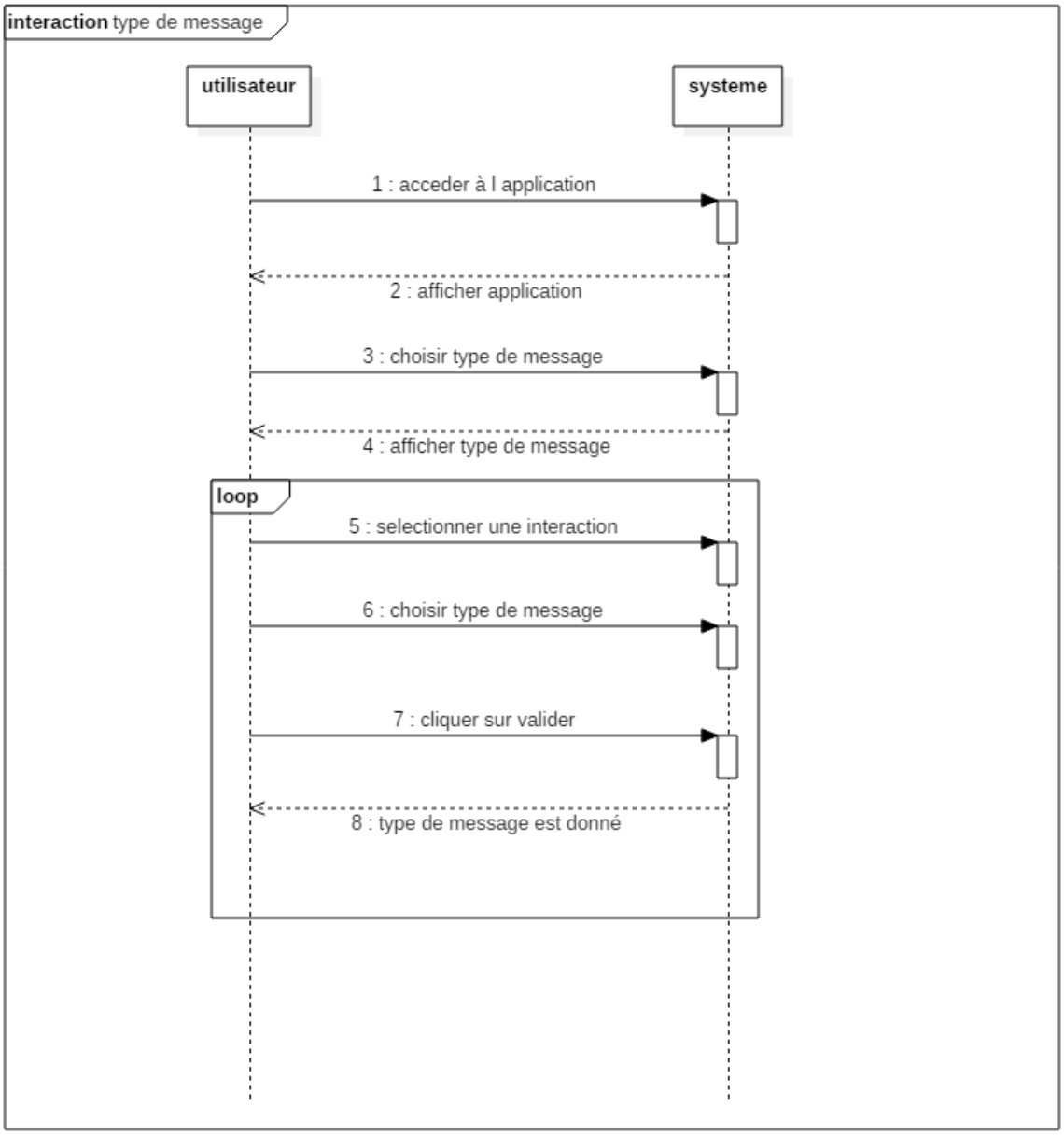


Figure 31: diagramme de séquence de haute niveau donner les type de message.

Description textuelle : analyser
Titre : analyserpadula
Acteur principale :

- Utilisateur.

Objectifs :

- Détecter les failles de modèle d'application selon lapropriété de politique de Padula.

Pré condition : Le cas d'utilisation commence lorsque :

-le diagramme d'application soit introduit.

-le politique de sécurité est introduire selon le modèle de sécurité MAC.

Enchainement nominal :

- L'utilisateur choisit analyser.
- L'utilisateur sélectionné padula.
- L'utilisateur sélectionné les deux propriétés.
- L'utilisateur validé.

-système cherche les failles.

-la liste des erreurs est affichée.

Enchainementalternative

- L'utilisateur choisit la propriété simple.

-l'enchainement commence de troisième étapes de scénario nominale.

- L'utilisateur choisit la propriété étoile

-l'enchainement commence de troisième étapes de scénario nominale.

Enchainementd'exception

- Diagramme de séquence non introduit et politique de sécurité non introduit ou choix de propriété non introduit.
- Scénario démarre après la 4°éme étape de scénario nominale.

-message d'erreur s'affiche.

Titre :Analyser biba

Acteur principale :

- Utilisateur.

Objectifs :

- Détecter les failles de modèle d'application selon la propriété de politique de biba.

Pré condition

- Le cas d'utilisation commence lorsque :

-le diagramme d'application soit introduit.

-le politique de sécurité est introduire selon le modèle de sécurité MAC.

Enchainement nominal :

- L'utilisateur choisit analyser.
- L'utilisateur sélectionne biba.
- L'utilisateur sélectionne les deux propriétés.
- L'utilisateur valide.

-système cherche les failles.

-la liste des erreurs est affichée.

Enchainement alternative

- L'utilisateur choisit la propriété simple.

-l'enchainement commence de troisième étapes de scénario nominale.

- L’utilisateur choisit la propriété étoile

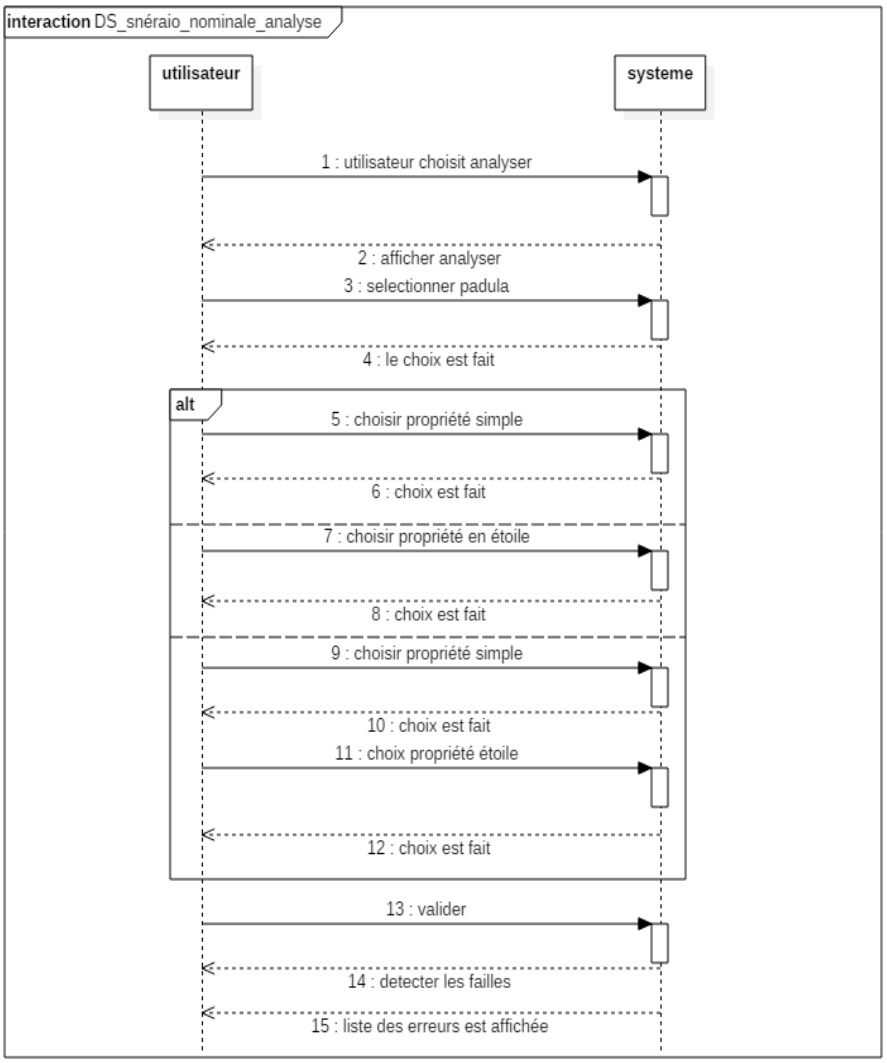
  -l’enchainement commence de troisième étapes de scénario nominale.

Enchainement d’exception

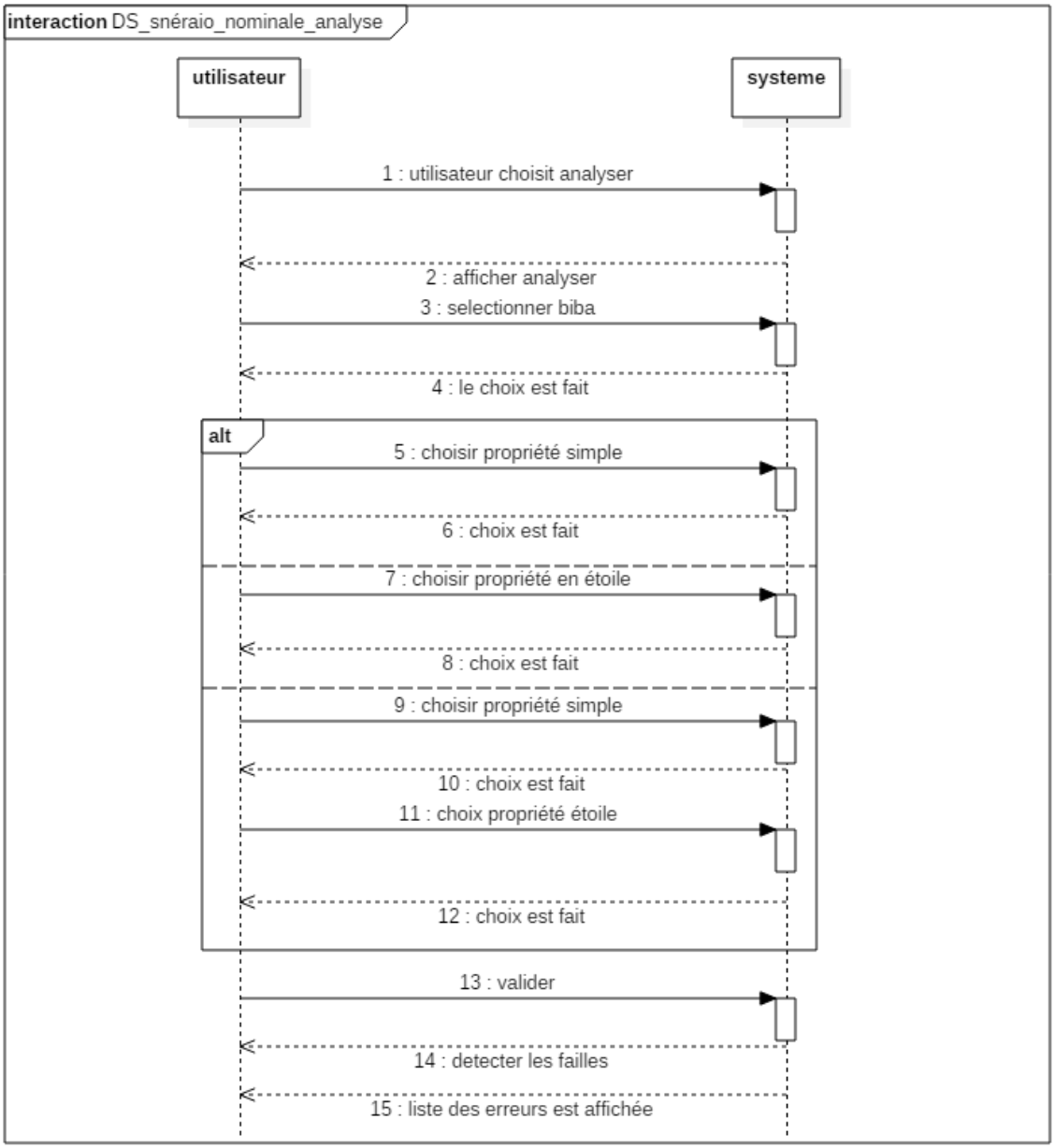
- Diagramme de séquence non introduit et politique de sécurité non introduit ou choix de propriété non introduit.

  - Scénario démarre après la 4°éme étape de scénario nominale.

  -message d’erreur s’affiche.



Figue 32: diagramme de haut niveau d’analyse selon padula



Figue 33: diagramme de haut niveau d'analyse selon biba

IV.2.3.Diagramme de classe d'analyse

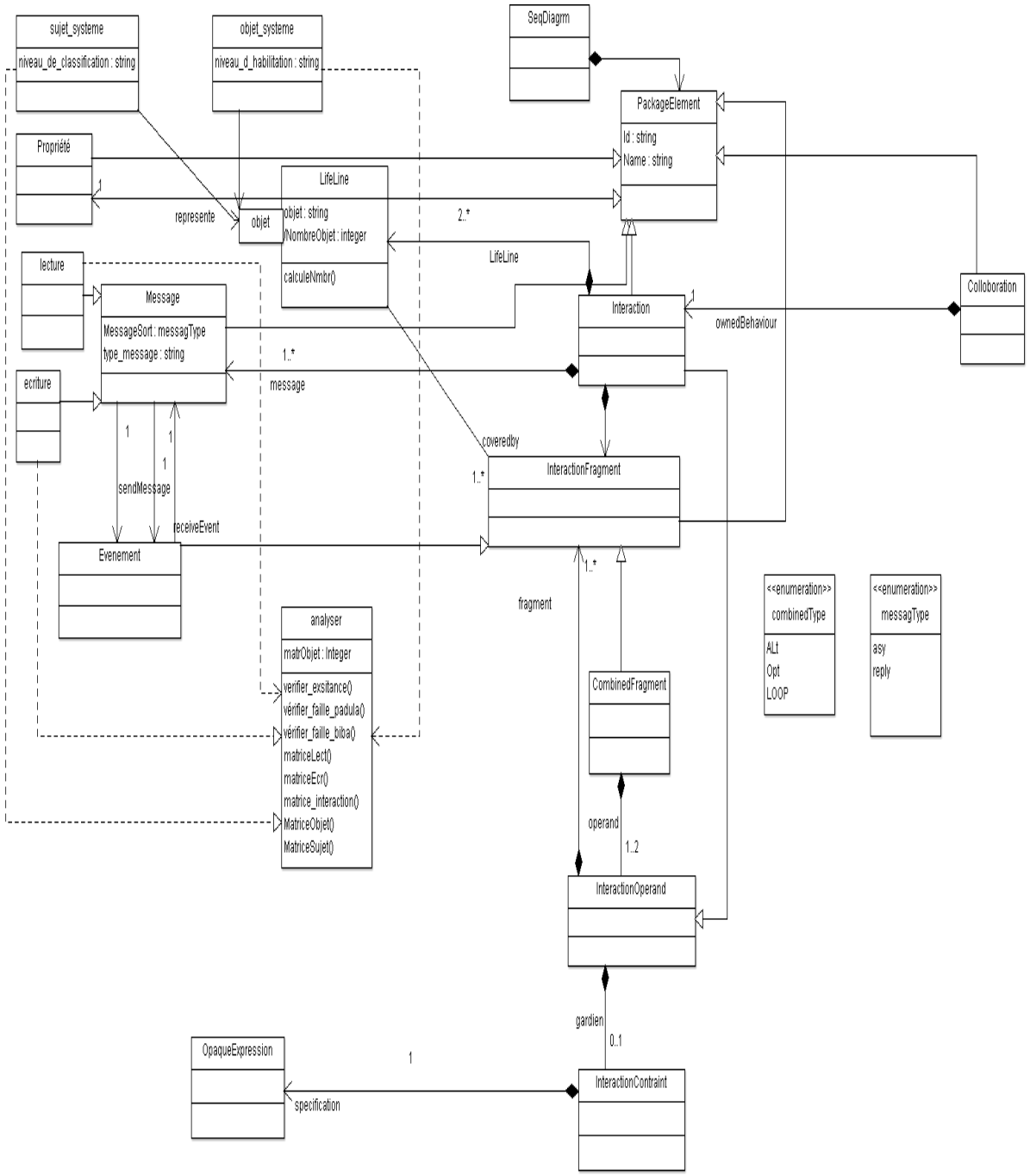


Figure 34: Diagramme de classe d'analyse de l'application

V. Algorithmes Transformant les diagrammes en matrices

1-Algorithme d'interaction de diagramme de séquence (modèle d'application) :

Cet algorithme permet de déduire la matrice d'interaction qui relie les objets (objet envoyé et objet reçue) avec des messages.

```
Fonction  matrice_interaction (matInte[][] : Entier , nombre_objet : Entier)

Var j : Entier

i : Entier

objenv, objrecu : Caractère

Début
  Pour i  de 0 à nombre_objet faire
     Pour j de 1 à nombre_objet faire
      Si (matInte[i][] =  objenvet matInte[][j]= objrecu) alors
                                    matInte[i][j] ← 1 ;
                Fin si
Fin pour
          Fin pour
```

2- L'algorithme utilisé pour la politique de contrôle d'accès :

Notre application offre la possibilité de saisir la politique de contrôle d'accès selon deux variantes : les sujets autorisé à accéder aux objets ou les sujets interdites à accéder aux objets. En se basant sur les niveaux de sécurités, ces deux variantes nous mènent vers deux matrices différentes : la matrice de sujet (regroupe les sujets et ces niveaux de sécurités) et la matrice d'objets (regroupe les objets et ces niveaux de sécurités).

2-1 Niveau de sécurité pour les sujets :

```
Fonction  matriceSujet (matSujet[][] : Entier, nombre_sujet : Entier, nombre_niveau : Entier)
Var j : Entier
 i : Entier
niveau : Caractère
objsele : Caractère
Début
Pour i de 1 à nombre_ niveau faire
          Pour j de 0 à nombre_sujet faire
                Si matSujet[i][] =  niveau  et matSujet[][j] = objsele alors
                                    matSujet[i][j] ←  1
                fin si


Fin pour
          Fin pour
```

2-2 Niveau de sécurité pour les objets:

```
Fonction  matriceObjet(matObjet[][] : Entier, nombre_objet : Entier, nombre_niveau :Entier)

Var j
 i : Entier
niveau : Caractère
objsele : Caractère
Début
Pour i de 1 à nombre_ niveau faire
        Pour j de 0 à nombre_objet faire
                Si matObjet[i][]= niveau matSujet[][j] = objsele alors
                                matObjet[i][j] ← 1
                fin si


Fin pour
        Fin pour
```

3-L'algorithme d'analyse pour la détection des failles:

```
Fonction verifier_faille_padula (matr[][] : Entier, lect : Entier ,ecrire : Entier)

Var

j : Entier

i : Entier

n1=-1 ;n2=-1 : Entier

objsele : chaine

pour i de 0 à lect faire

        pour j de 1 à lect faire

          si (matr[i][j] != 0 et matr[i][j] =1) alors

        matriceSujet(matr[][], i ,objSel)

                        n1←objSel

        matriceObjet(matr[][], j , objSel)

                        n2←objSel
```

si (n1>= n2)

Ecrire (« aucune faille est détecter »)

fin si

sinon

Ecrire (« une faille est détecter »)

Sinon

si (matr[i][j] != 0 et matr[i][j] =1) alors

matriceSujet(matr[][], i , objSel)

n1←objSel

matriceObjet(matr[][], j , objSel)

n2←objSel

si (n1<= n2)

Ecrire (« aucune faille est détecter »)

fin si

sinon

Ecrire (« une faille est détecter »)

fin pour

fin pour

Cet algorithme pour la détection des failles par rapport le modèle de padula selon les deux propriétés (propriété simple et propriété étoile) il fait l'appelle à la matrice de sujet et la matrice d'objet.

VI. Exemple des diagrammes de séquence

VI.1. Exemple1

Cet exemple figure 35montre un diagramme de séquence contenant trois objets (objet 1, objet 2 et objet 3).

La base de notre politique de sécurité et de choisir MAC :

- qui est le sujet et qui est l'objet ?
- quels sont les niveaux de sécurité pour le sujet et l'objet ?
- quelle est le type de message interchangeable entre le sujet et l'objet ?

Figure ci-dessus illustre les défirent base de politique de sécurité MAC pour notre exemple :

Nom	Partitionnement S/O	Niveau de sécurité	Type message
Objet 1	Sujet	confidentiel	o1-> o2 lecture
Objet 2	Sujet	public	o2->o3écriture
Objet 3	Objet	Top secret	o3->o2 lecture o3->o2écriture

Tableau 1:politique de sécurité de l'exemple 1

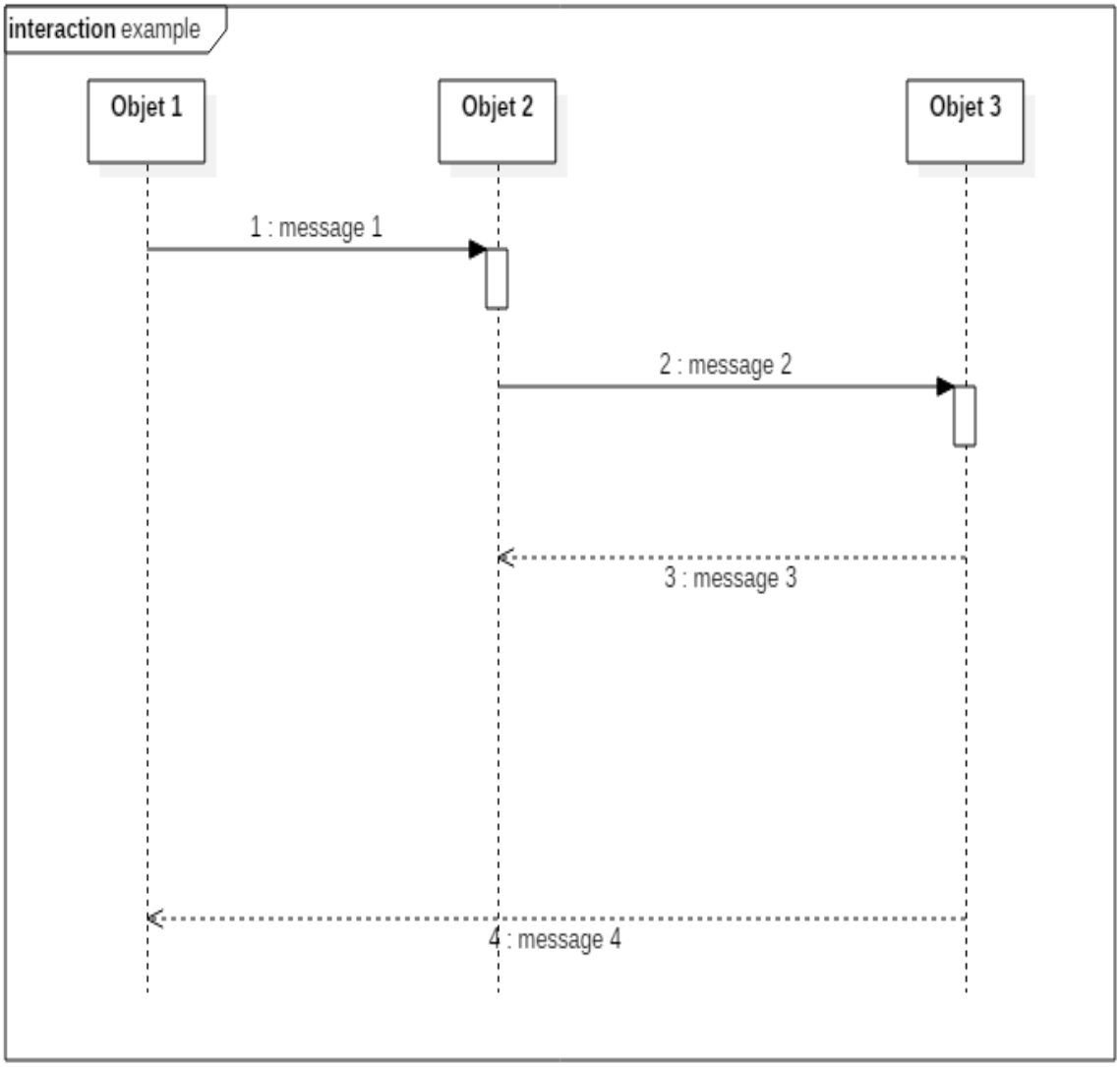


Figure 35: exemple 1 diagramme de séquence

1-Algorithme d'interaction de diagramme de séquence (modèle d'application) :
Après avoir appliqué l'algorithme Fonction matrice_interaction (matInte[][] : Entier , nombre_objet : Entier) sur l'exemple 1, on déduit la matrice d'interaction. Figure montre le résultat du déroulement.

Nom	Objet 1	Objet 2	Objet 3
Objet 1	0	1	0
Objet 2	0	0	1
Objet 3	1	1	0

Tableau 2: matrice d'interaction

2- L'algorithme utilisé pour la politique de contrôle d'accès :

2-1 Niveau de sécurité pour les sujets :

Apres avoir appliqué l'algorithme matrice de sujet on a introduit la figure ci-dessus :

Nom	Top secret	secret	confidentiel	public
Objet 1	0	0	1	0
Objet 2	0		0	1

Tableau 3: les niveaux pour les sujets

2-2 Niveau de sécurité pour les objets:

Apres avoir appliqué l'algorithme matrice d'objet on a introduit la figure ci-dessus :

Nom	Top secret	secret	confidentiel	public
Objet 3	1	0	0	0

Tableau 4: Figure : les niveaux pour les objets

3-L'algorithmes d'analyse pour la détection des failles :

Dans cet exemple une faille est détectée entre sujet (objet 2) et l'objet (objet 3).

Cette détection est dû au faite le sujet peut accéder au l'objet par lecture, alors la politique lui interdite.

VI.2. Exemple 2

Cet exemple figure 36 montre un diagramme de séquence contenant trois objets (publiphone, standard et utilisateur).

Figure ci-dessus illustre les défirent base de politique de sécurité MAC pour notre exemple :

nom	Partitionnement S/O	Niveau de sécurité	Type message
utilisateur	Sujet	Top secret	1 :utilisateur ->publiphone lecture 2 : utilisateur -> publiphone écriture 5 : utilisateur -> publiphone écriture 11: utilisateur -> publiphone lecture 13: utilisateur -> publiphone lecture 1 : utilisateur -> publiphone lecture
publiphone	Objet	secret	3 : Publiphone-> publiphone écrire 4 : Publiphone-> publiphone lecture 6 : publiphone -> standard écriture 10: Publiphone-> utilisateur écrire 12: Publiphone-> standard écrire 15 : Publiphone-> publiphone écrire 17 : Publiphone-> publiphone écrire
standard	Sujet	public	7: standard -> publiphone lecture 8 : standard ->publiphoneécriture 9 : standard -> publiphone écriture

Tableau 5: politique de sécurité de l'exemple 2

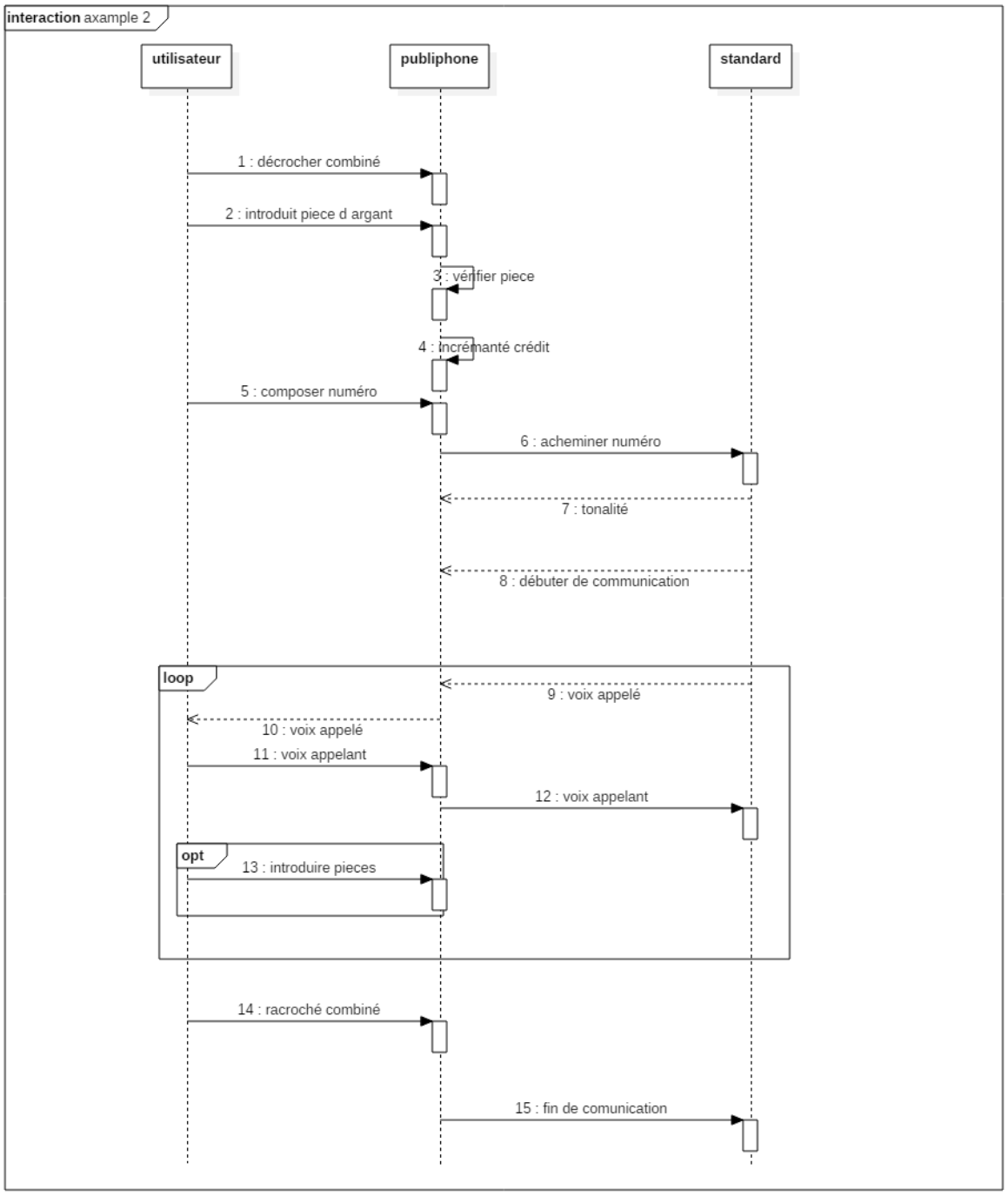


Figure 36: Exemple 2 sur diagramme de séquence

1-Algorithme d'interaction de diagramme de séquence (modèle d'application) :

Après avoir appliqué l'algorithme Fonction matrice_interaction (matInte[][] : Entier , nombre_objet : Entier) sur l'exemple 2, on déduit la matrice d'interaction. Figure montre le résultat du déroulement.

nom	utilisateur	publiphone	standard
utilisateur	0	1	0
publiphone	1	0	1
standard	0	1	0

Tableau 6: matrice d'interaction

2- L'algorithme utilisé pour la politique de contrôle d'accès :

2-1 Niveaux de sécurité pour les sujets :

Apres avoir appliqué l'algorithme matrice de sujet on a introduit la figure ci-dessus

nom	Top secret	secret	confidentiel	public
utilisateur	1	0	0	0
standard	0	0	0	1

Tableau 7: les niveaux pour les sujets

2-2 Niveau de sécurité pour les objets:

Apres avoir appliqué l'algorithme matrice d'objet on a introduit la figure ci-dessus :

nom	Top secret	secret	confidentiel	public
publiphone	0	1	0	0

Tableau 8: les niveaux les objets

3-L'algorithmes d'analyse pour la détection des failles :

Dans cet exemple une faille est détectée entre standard et publiphone dans le cas de propriété simple.

Cette détection est dû au faite le sujet peut accéder au l'objetpar lecture, alors la politique lui interdite

VII. Conception

VII.1. Diagrammede classe de conception

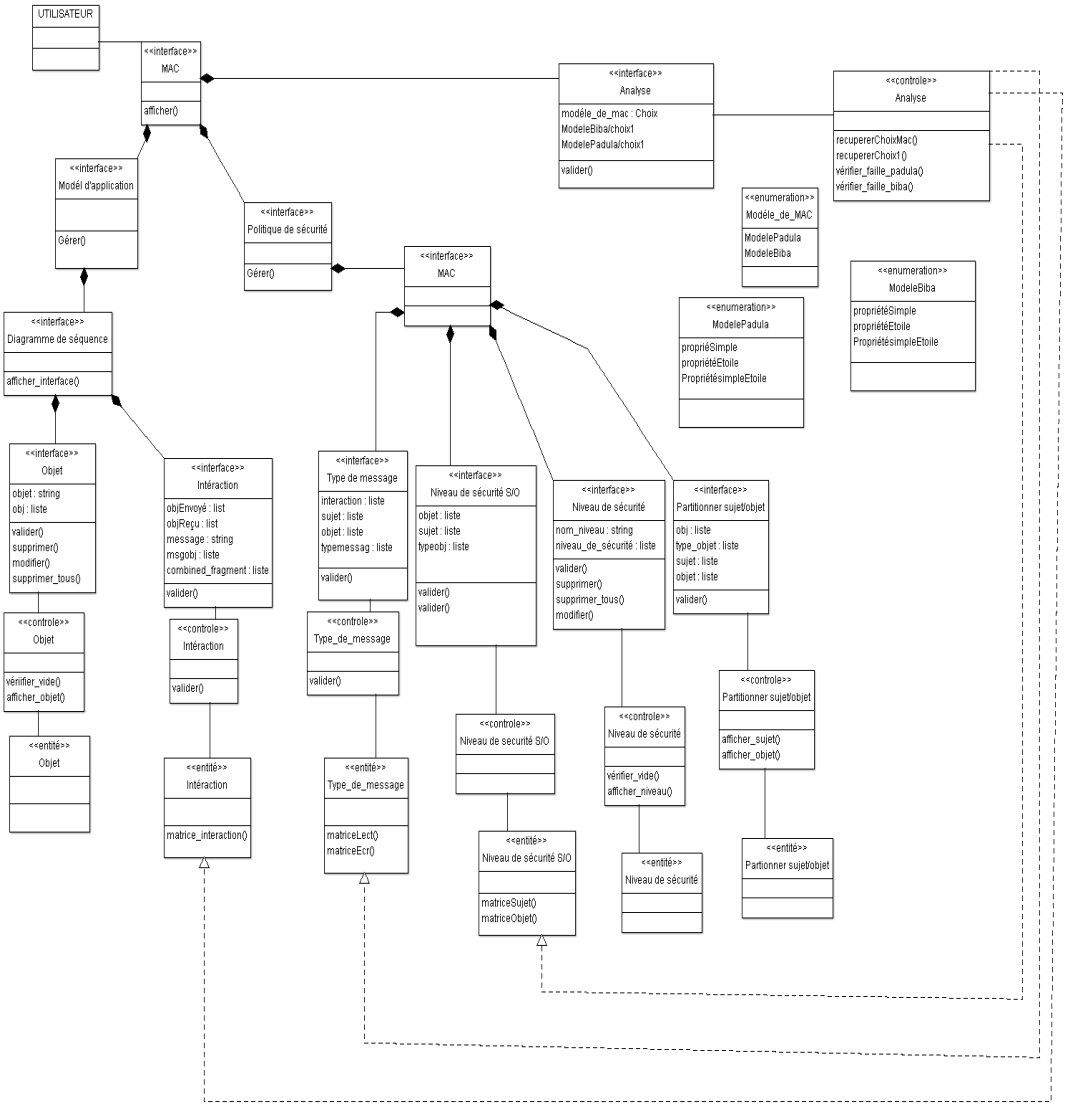


Figure 37: Diagramme de classe de conception de l'application

VII.2. Diagramme de séquence de conception de l'application

VII.2.1. Diagramme de séquence gérer modèle d'application

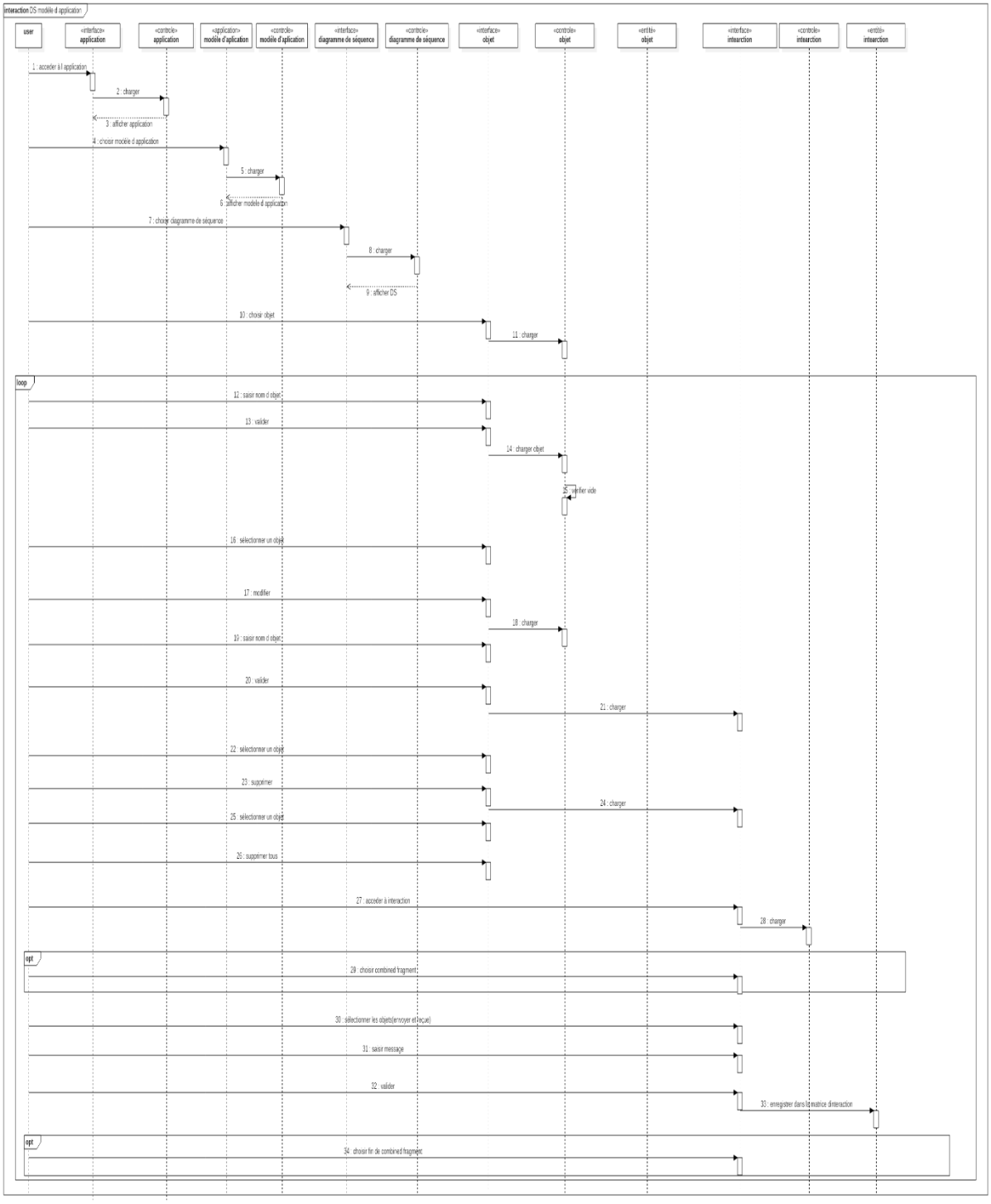


Figure 38: Diagramme de séquence de modèle d'application

VII.2.2. Diagramme de séquence gérer politique de sécurité

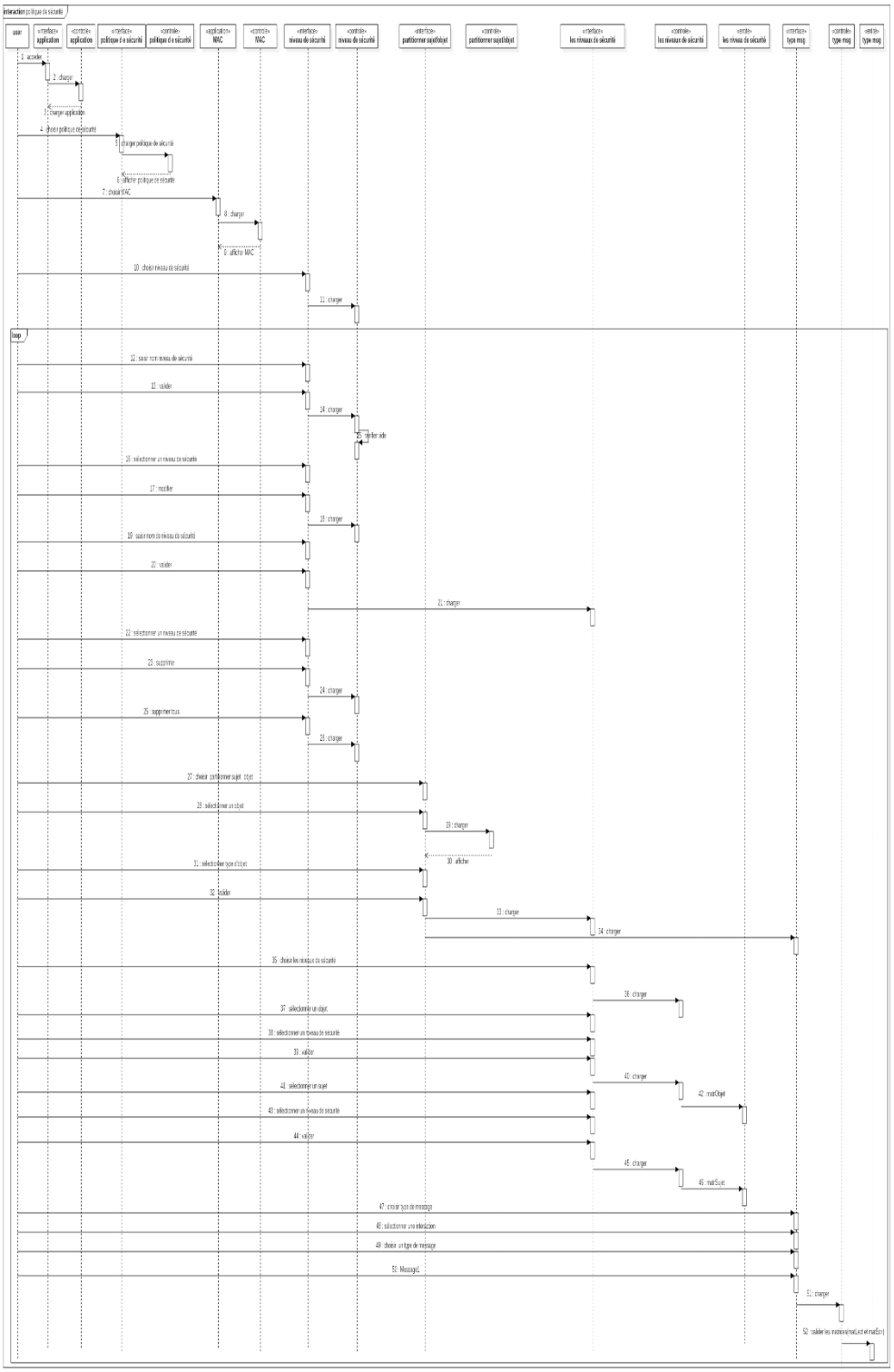


Figure 39: Diagramme de séquence de modèle d'application

VII.2.3. Diagramme de séquence d'analyse

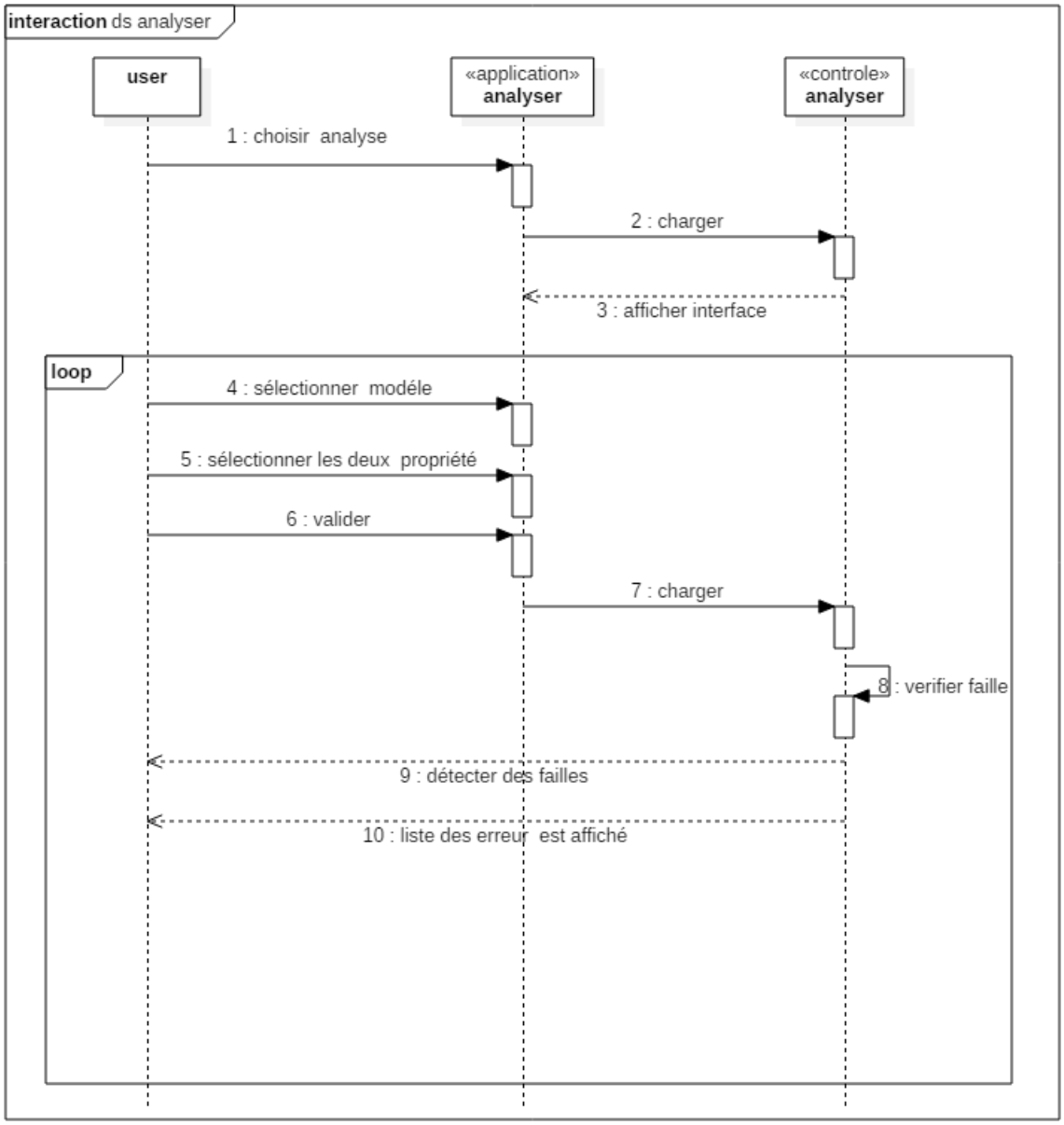


Figure 40: Diagramme de séquence d'analyse

VIII. Diagramme d'activité de l'application

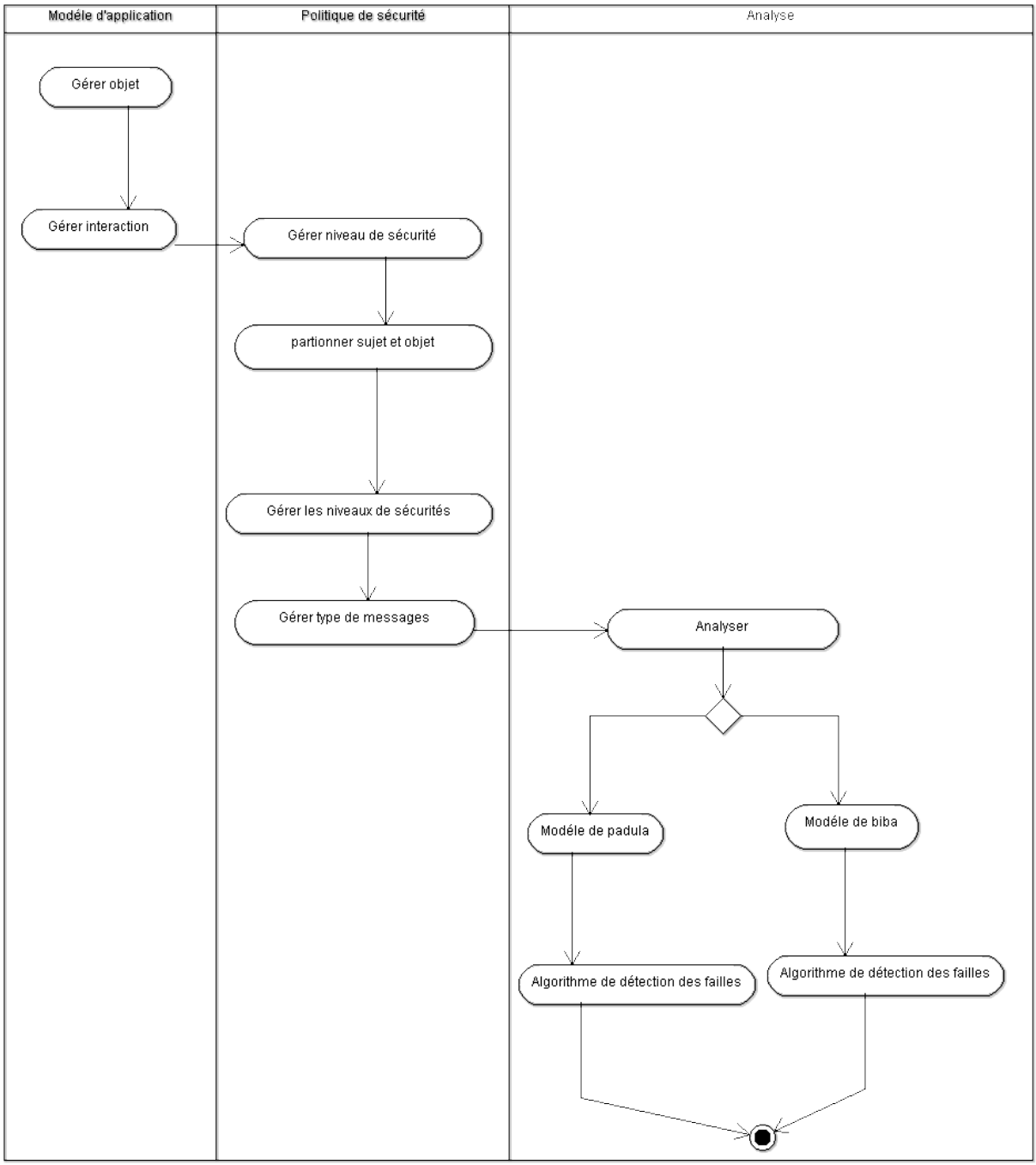


Figure 41: Diagramme d'activité d'application

IX. 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.

Chapitre 4 : Implémentation

Introduction

Nous abordons dans ce chapitre la partie implémentation qui mettra en pratique la conception de notre modèle de contrôle d'accès MAC. 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.

I. 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.

II. Pourquoi avons-nous choisi le langage JAVA ?

- L'avantage principal de Java par rapport aux autres langages c'est sa portabilité, le faitqu'un programme Java puisse théoriquement être exécuté sur n'importe quelle plate-forme.**[11]**
- Il s'agit d'un langage de conception très performant qui a été adopté par la majorité des fournisseurs
- **i**ntégrées de sécurité offrent un sentiment de confiance aux programmeurs comme aux utilisateurs des applications.
- Java est assurément un bon langage de programmation. Il s'agit, sans aucun doute, de Fun des meilleurs disponibles pour un programmeur sérieux.

III. Environnement de développement

C'est un environnement de développement intégré (EDI) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, éditeur graphique d'interfaces et de pages Web). Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X et Open VMS. **[12]**

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 : 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.

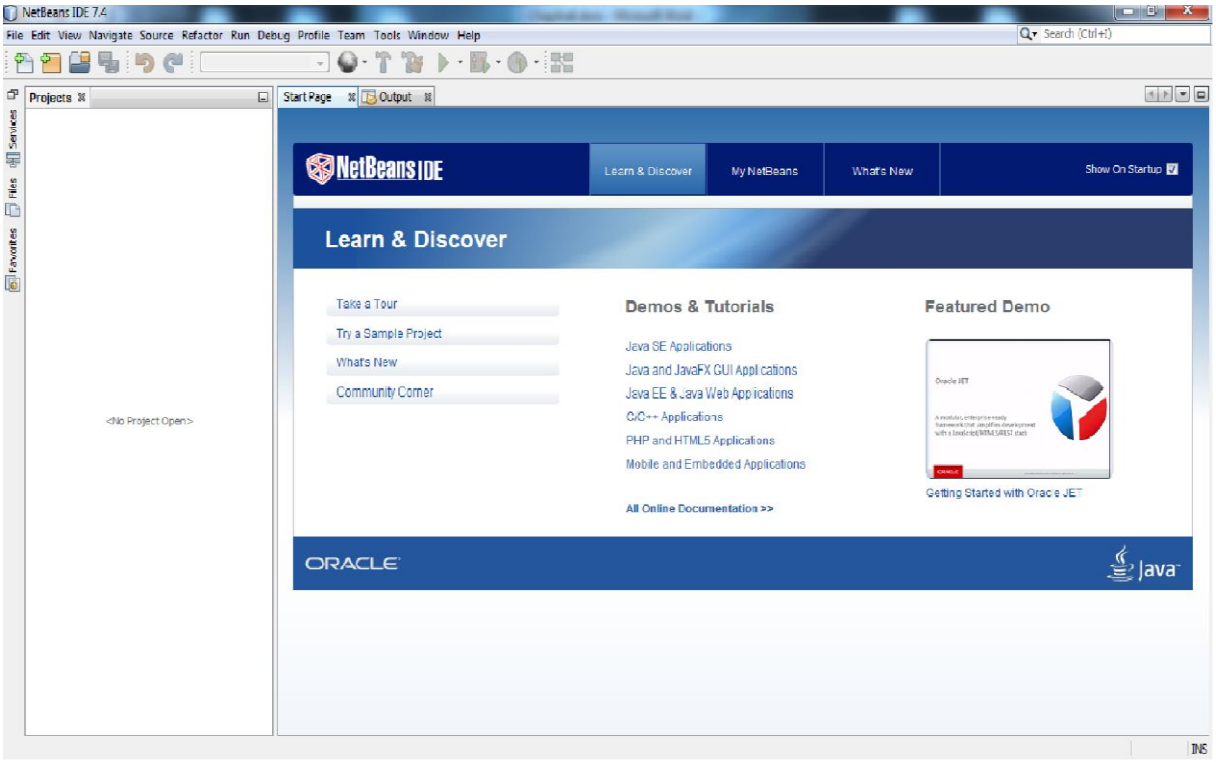


Figure 42: Interface de l'IDE NetBeans

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

Dans ce qui va suivre nous allons proposer code source en java d'algorithmes décrit dans le chapitre précédent, algorithme de détection des faille selon padula.

```
public void verifier_faille_padula(){

int i1=-1,i2=-1;

if (propeiété_simple.isSelected())

{

for(inti=0; i<i++){

for(j=1; j<colomn.size;j++){

if (tableModel.getValueAt(i, j) != null && (Integer) tableModel.getValueAt(i, j) == 1) {

for (int a = 1; a <ableModel.getColumnCount(); a++) {

if (ableModel.getValueAt(i, a) != null && (Integer) ableModel.getValueAt(i, a) == 1)

        {

            i1 = a;

         }

                                                                    }
```

```
for (int a = 1; a <tbleModel.getColumnCount(); a++) {
        if (tbleModel.getValueAt(j- 1, a) != null && (Integer) tbleModel.getValueAt(j - 1,   a)
        == 1) {
                        i2 = a;
                                                }
                                                                }
if (i1 >= i2) {
        jTextArea1.append(tableModel.getValueAt(i,0) + " et " +
        tableModel.getColumnName(j) + " aucune faille estdétecter\n");
          } else {
jTextArea1.append(tableModel.getValueAt(i, 0) + " et " +  tableModel.getColumnName(j) +
"une faille estdétecter\n");
                }
              }
           }
         }
      }
else if (jRadioButton5.isSelected()) {
for (inti = 0; i<tableodel.getRowCount(); i++) {
for (int j = 1; j <tableodel.getColumnCount(); j++) {
if (tableodel.getValueAt(i, j) != null && (Integer) tableodel.getValueAt(i, j) ==   1) {
for (int a = 1; a <ableModel.getColumnCount(); a++) {
if (ableModel.getValueAt(i, a) != null && (Integer) ableModel.getValueAt(i, a) == 1)   {
i1 = a;
                                                                }
 }
for (int a = 1; a <tbleModel.getColumnCount(); a++) {
if (tbleModel.getValueAt(j - 1, a) != null && (Integer) tbleModel.getValueAt(j - 1, a) == 1) {
                        i2 = a;
```

```
}

}

if (i1 <= i2) {

jTextArea1.append(tableodel.getValueAt(i, 0) + " et " + tableodel.getColumnName(j) + "
aucune faille estdétecter\n");

                } else {

jTextArea1.append(tableodel.getValueAt(i, 0) + " et " + tableodel.getColumnName(j) + "  une
faille estdétecter\n");

                }

              }


            }

          }


      }

}
```

V. Implémentation et teste de l'application

Notre application est composée de deux parties, la première partie consiste à introduire le modèle d'application (diagramme de séquence) 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 (objet, interaction) et un autre module pour la détection des défiances de sécurité du modèle d'application.

Pour le teste nous avons choisi l'exemple 2 décrit dans le chapitre.

VI. Présentation des interfaces de notre application

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

VI.1. Interface de modèle d'application (gérer objet)

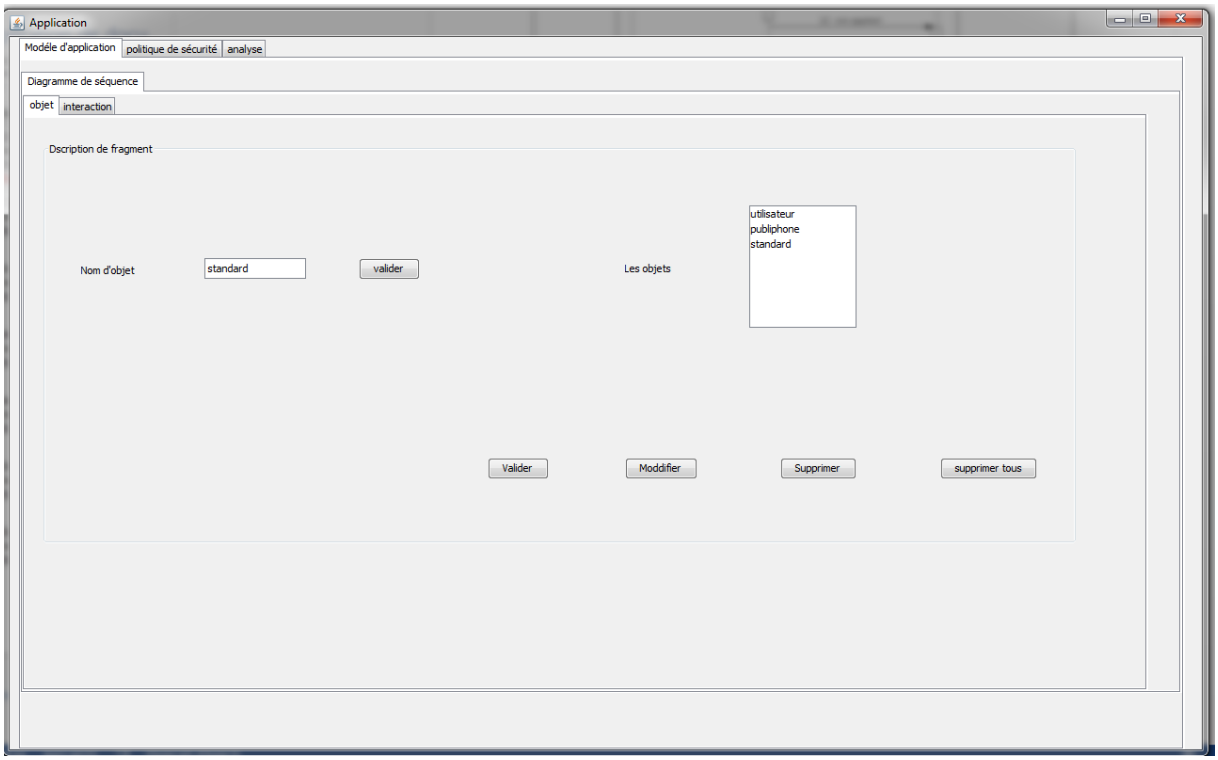


Figure 43: Interface modèle d'application (Gérer d'objet)

VI.2. Interface de modèle d'application (gérer interaction)

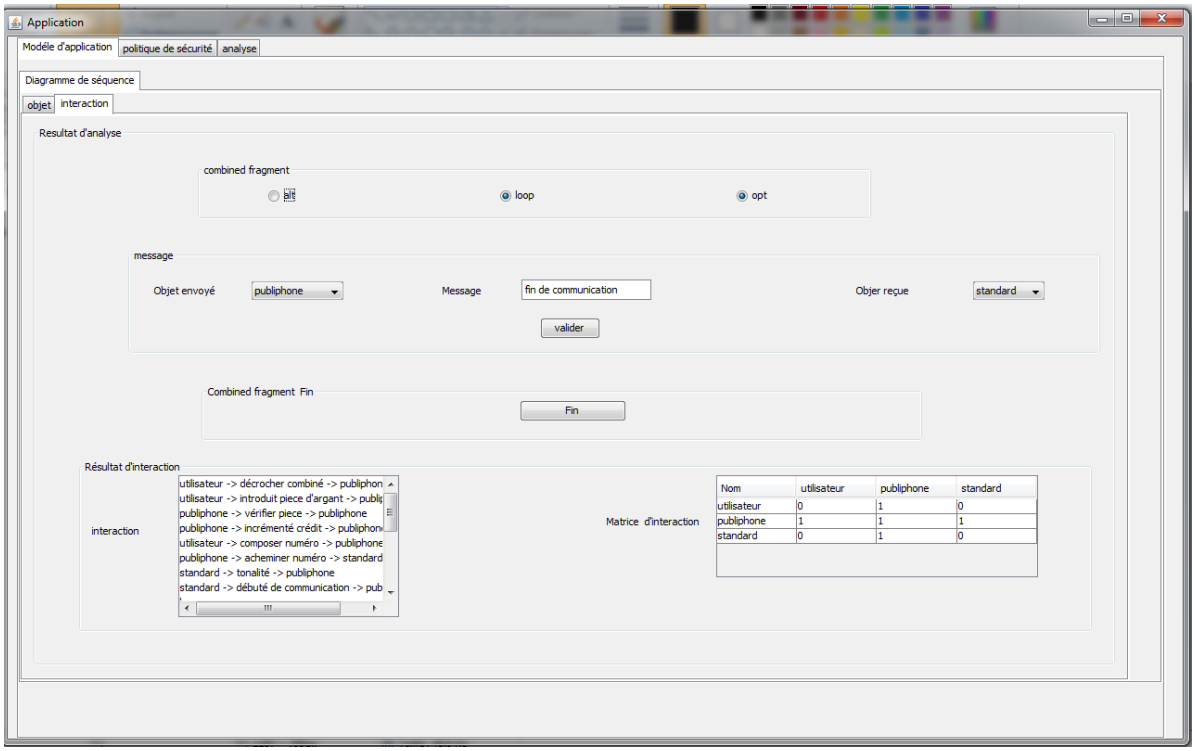


Figure 44: Interface modèle d'application (Gérer interaction)

VI.1. Interface de la politique de sécurité (gérer niveau de sécurité)

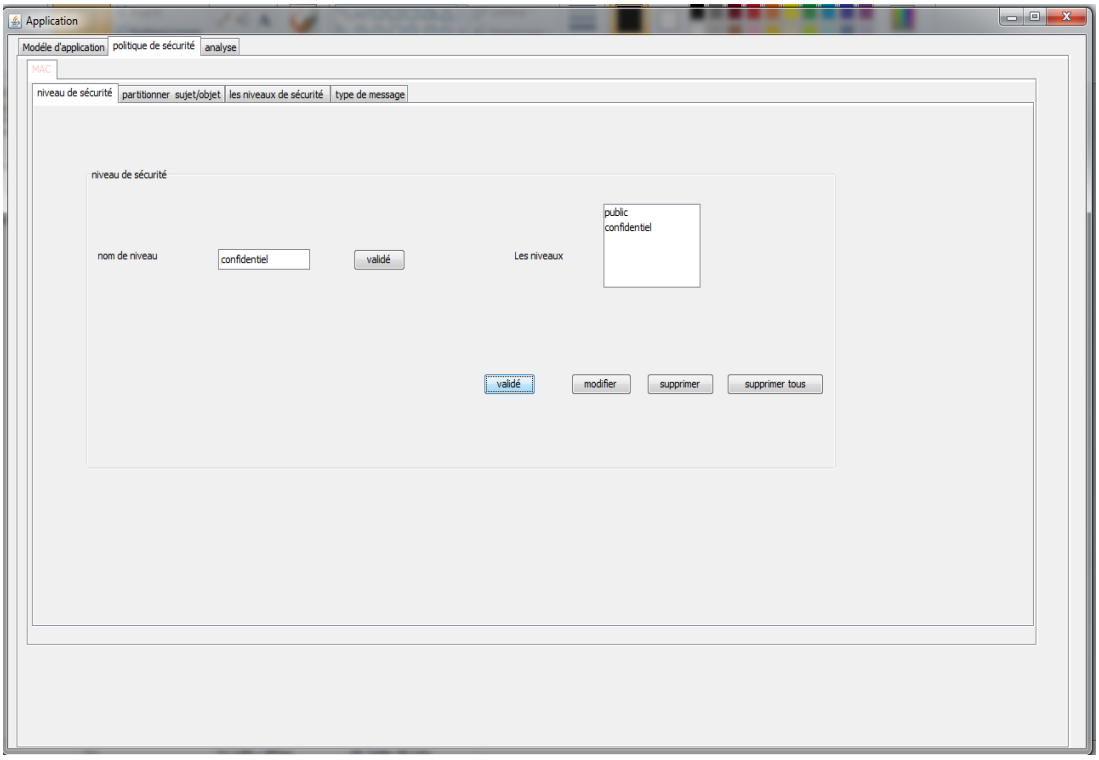


Figure 45: Interface Politique de sécurité (Gérer niveau de sécurité)

VI.1. Interface d'analyse

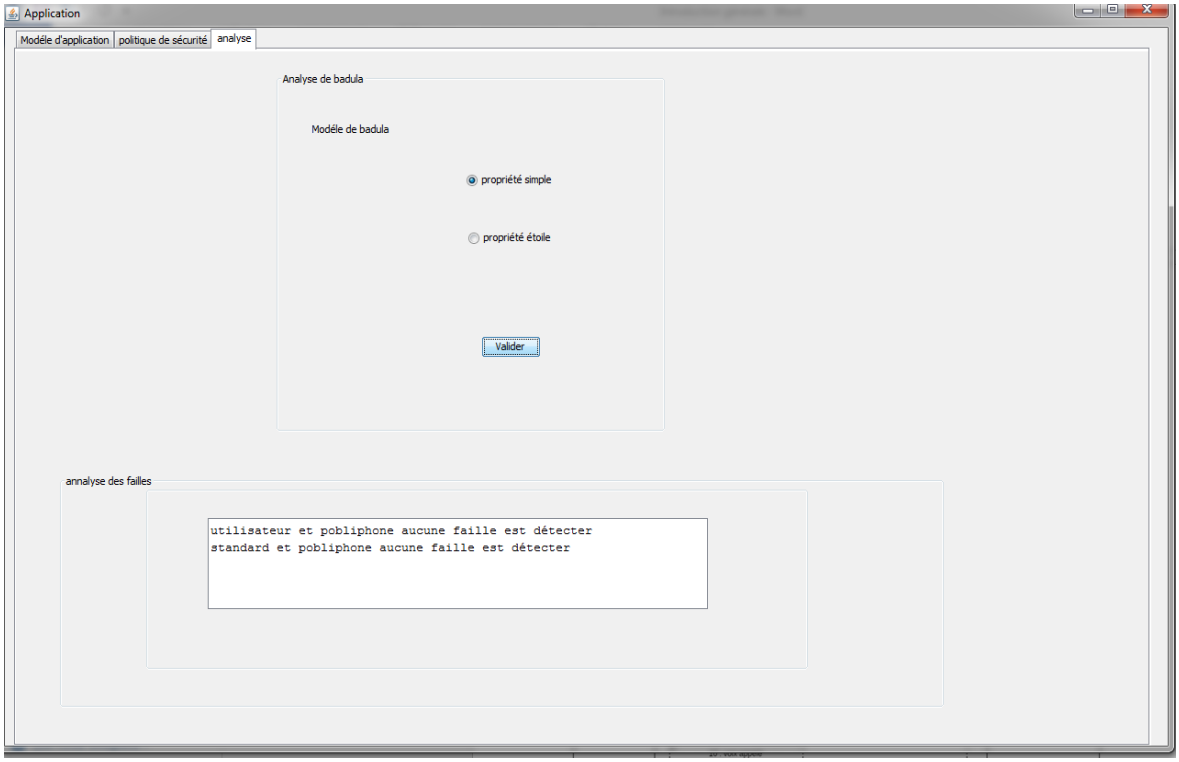


Figure 46: Interface d'analyse

I. Conclusion

Dans ce chapitre nous avons commencé par définir le langage java suivi d'implémentation de quelque algorithme. Et enfin nous avons testé les exemples décrit dans le chapitre précèdent.

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.

Afin 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 MAC, ces données sont synthétisées dans des matrices adéquates, pour cela on a proposé des algorithmes qui nous a permis de détecter les failles.

Toute fois ce travail ne permet pas de détecter toutes les failles connues dans le domaine de la sécurité.

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)

Bibliographies

[1]Donald L. Pipkin ; Sécurité des systèmes d'information ;CampusPress, 2000.

[2]Archimbaud J.L, Longeon R.; Guide de la sécurité des systèmes d'information à l'usage des directeur, 1999

[3] Bäkeofe JayantD, Ubale Swap naja A, modanai Dattatray G, Apte Sulabha S PhD; Analysis of DAC MAC RBAC Access Control based Models for Security; Octobre 2014; international Journal of Computer Applications (0975 – 8887) Volume 104 – No.5

[4]Peter Y. A. Ryan; Mathematical Models of Computer Security; 2001 ;R. Focardi and R. Gorrieri (Eds): FOSAD 2000, LNCS 2171.

[5]OMG;OMG Unified Modeling Language TM (OMG UML)**;** 2015-03-01.

[6]Chao Li, Liang Dou and Zongyuan Yang; A metamodeling level transformation from UML sequence diagrams to Coq; East China Normal University Shanghai, China ; (No.61070226).

[7] Xavier Blanc, Isabelle Mounier ;UML2 pour les développeurs, Ed. Eyrolles ;2006.

[8]Pascal Roques (2004) UML 2 par la pratique, Eyrolles, 311 p

[9] DebasishKundu, Monalisa Sarma, DebasisSamanta; A UML model based approach to detectinfeasiblepaths; 2015; 0164-1212 Elsevier Inc.

[10] Philippe Kruchten. The Rational Unified Process An Introduction, Second Edition.Addison Wesley, 2000.

[11] S. Laporte; Introduction en java; Lycée Louise Michel BTS IG DA.

[12] Oracle**;** NetBeans; https://netbeans.org/index_fr.html ; 24-05-2017.