République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention d'un diplôme de Master en informatique.
Option : Ingénierie du logiciel.

Thème :

Délégation non monotone dans le contrôle d'accès

Présenté par :
Bouafia Doria .
Ziraoui Bouchra.

Sujet proposé et encadré par :
Mlle N.Boustia,Doc

Soutenu le 01 juillet 2013 devant le jury :

Mme Abed.H	Présidente
Mme Fareh.M	Examinatrice
Mme Mancer.Y	Examinatrice
Mme Boustia.N	Rapporteur

2012/2013

Délégation non monotone dans le contrôle d'accès

Résumé :

Cette étude a pour but la réalisation d'un modèle de contrôle d'accès basé sur la logique de description défaut (δ) et exception(ε)pour représenter le contexte en intégrant les principaux mécanismes d'inférence.

Nous avons appelé notre modèle DELEGATION_OrBAC δε ,il est inspiré du modèle OrBAC comprenant la modélisation du concept de délégation .L'attribution d'autorisation à un utilisateur se fait suivant son rôle dans l'organisation et selon le contexte, cette attribution d'autorisation peut aussi résulter d'une délégation de la part d'un autre utilisateur de l'organisation , cette délégation a plusieurs types et les droits transmis à travers son attribution peuvent être révoqués de plusieurs manières également.

L'avantage principal de notre modèle est la représentation de contexte et la déduction de nouvelles autorisations selon ce contexte.

Mots clés : OrBAC, Delegation , logique non monotone , logique de description .

Abstract :

This study aims to achieve a model of access control based on description logic including default (δ) and exception (ε) operators to represent the context and with integrating inference mechanisms.

We called our model DELEGATION_OrBAC δε.It is inspired of OrBAC model including the modelisation of delegation concept. Assigning authorization to a user is following his role in the organization and according to the context of this authorization, the authorisation can also result from a delegation of permission from another user in the organization, this delegation has different types and rights transmitted through its can be revoked in different ways as well.

The main advantage of our model is the representation of the context and the deduction of new authorizations according to this context.

Key words: OrBAC,Delegation, non monotonic logic, description logic.

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Table des figures

Introduction générale

Dans un premier temps, nous établiront une étude sur le modèle de contrôle d'accès OrBAC. On s'intéressera par la suite au concept de délégation ainsi qu'aux logiques de description en général. Une étude de la logique CLASSIC$_{\delta\varepsilon}$ sera faite également et nous terminerons par la modélisation et la réalisation de l'outil simulant notre modèle. Plus de détails dans ce qui suit :

Chapitre 1: Etude du modèle de contrôle d'accès OrBAC

Dans ce chapitre on s'intéresse à l'étude du modèle OrBAC en donnant les détails sur les entités et les relations qui le composent et quelques exemples d'illustrations.

Chapitre 2: Etude des délégations dans le modèle OrBAC

Dans ce chapitre on s'intéresse à l'étude du concept de délégation dans le modèle OrBAC ainsi qu' au concept de façon générale en détaillant ses types et en donnant des exemples .

Chapitre 3: Etude des Logiques de Description

Dans ce chapitre on présente les LDs classiques en donnant principalement leur présentation, leurs langages (terminologique et assertionnel),et leurs mécanismes d'inférence .

Chapitre 4: Etude de la logique CCLASSICδε

Dans ce chapitre on présente la logique CCLASSICδε en détaillant principalement sa syntaxe, sa sémantique.

Chapitre 5: Modèle de contrôle d'accès DELEGATION-OrBACδε

Dans ce chapitre on conceptualise le DELEGATION-OrBACδε, on présente les différents axiomes qui permettent de construire notre base de connaissances LD avec l'usage de défaut (δ) et exceptions (ε).

Chapitre 6: Réalisation du modèle DELEGATION-OrBACδε

Ce chapitre a été consacré aux évaluations du modèle **DELEGATION-OrBACδε**, on va faire une description des principales étapes de l'étude en les illustrant à l'aide des interfaces graphiques à travers un exemple d'illustration qui n'est autre qu'un système d'information médical. Ainsi que la présentation des différents outils et environnements de la réalisation.

Chapitre 01

Etude du modèle de contrôle d'accès OrBAC .

Etude du modèle de contrôle d'accès OrBAC

Chapitre 01

1. Introduction :

Les modèles de contrôle d'accès comme DAC, MAC, RBAC, TBAC ou TMAC ne permettent de modéliser que des politiques de sécurité qui se restreignent à des permissions statiques. Ils n'offrent pas la possibilité d'exprimer des règles contextuelles relatives aux permissions, aux interdictions, aux obligations et aux recommandations. Ce type de règle est particulièrement utile pour exprimer des politiques de sécurité comme par exemple dans le domaine médical. D'où l'apparition d'un nouveau modèle qui permet de spécifier de telles politiques de sécurité contextuelles. Ce modèle appelé Organisation Based Access Control (ORBAC) s'appuie sur un langage formel basé sur la logique du premier ordre. **[1]**
Nous allons dans ce chapitre détailler tous les aspects du modèle ORBAC.

2. Le modèle ORBAC :

Le modèle OrBAC définit un certain nombre d'entités et de relations. (**Figure 1.1)**

On voit se dessiner la structure à deux niveaux de la politique de sécurité :

- ✓ Un niveau concret : sujet, action, objet.
- ✓ Un niveau abstrait : rôle, activité, vue.

Le sujet est mis en correspondance avec le rôle, l'objet avec la vue et l'action avec l'activité. On remarquera également la position centrale de l'organisation et la présence du contexte. **[1]**

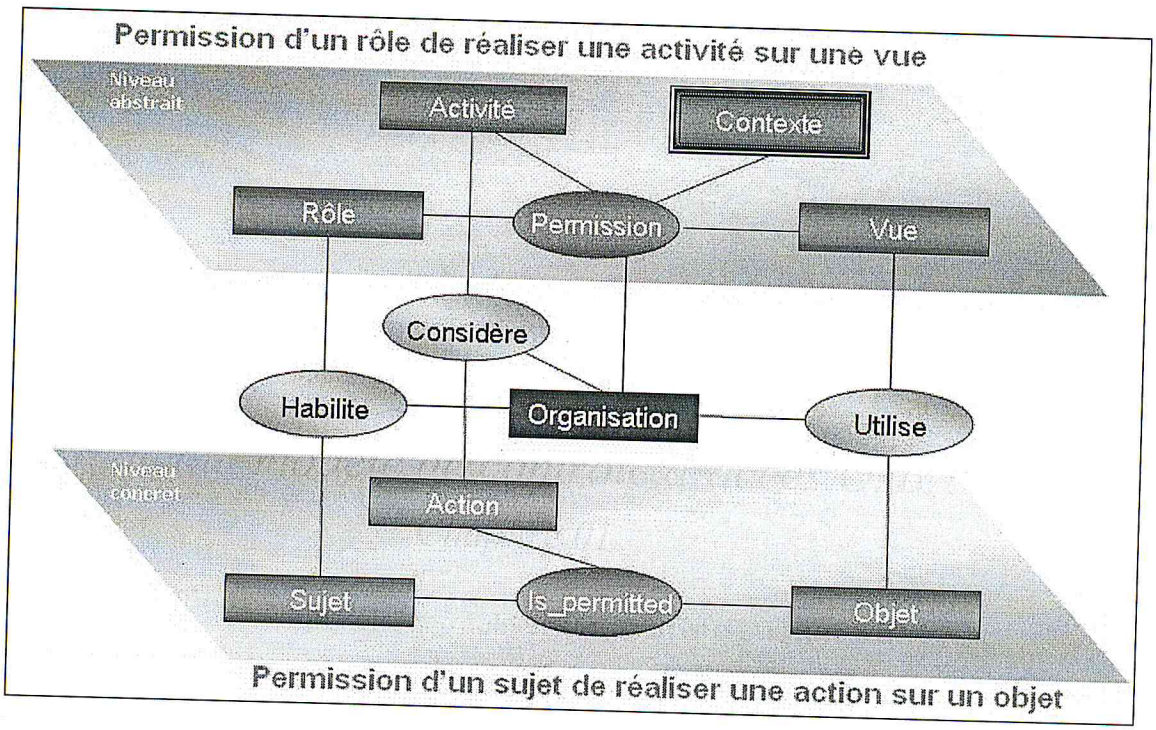

Figure 1.1:la structure du modèle OrBAC.**[3]**

2.1. Les éléments du modèle

Dans cette section, nous détaillerons tous les éléments du modèle OrBAC :

2.1.1 .L'organisation :

L'entité principale et centrale dans le modèle OrBAC est le concept d'organisation. Une organisation peut être vue comme un groupe organisé d'entités actives, c'est-à-dire de sujets jouant certains rôles. Notons qu'un groupe de sujets n'est pas nécessairement considéré comme une organisation. Autrement dit, le fait que chaque sujet joue un rôle dans l'organisation correspond à certain accord entre les sujets pour former une organisation. **[1]**

Par exemple : dans le domaine médical « Service de Cardiologie et médecine interne de l'Hôpital Frantz Fanon », « Service de neurologie de l'Hôpital Frantz Fanon », « Service de Psychiatrie de l'Hôpital Frantz Fanon », « Direction de l'Hôpital Frantz Fanon»… etc.

2.1.2 .Les sujets et les rôles :

L'entité Sujet est utilisée différemment selon les modèles de sécurité. Dans le modèle OrBAC, un sujet peut être soit une entité active, c'est-à-dire un utilisateur, soit une organisation. **[1]**

Par exemple, "Leila" , Bouchra" , "Doria" , etc., peuvent être des sujets, tout comme les organisations " département comptable de la clinique privée Lilia" , " le service des urgences de l' hôpital Franz Fanon" , etc.

Dans le modèle OrBAC, l'entité Rôle est utilisée pour structurer le lien entre les sujets et les organisations **[1]** . Dans le domaine médical, les rôles " cardiologue" , " infirmière" ou "médecin" , sont joués par des utilisateurs alors que les rôles " service des urgences" ou " unité des soins intensifs" sont joués par des organisations. Comme les sujets jouent des rôles dans des organisations, OrBAC introduit une relation entre ces entités : la relation Habilite (**Figure 1.2**). Habilite(org, s, r) signifie que org habilite le sujet s à jouer le rôle r en considérant org une organisation, s un sujet ,et r est un rôle

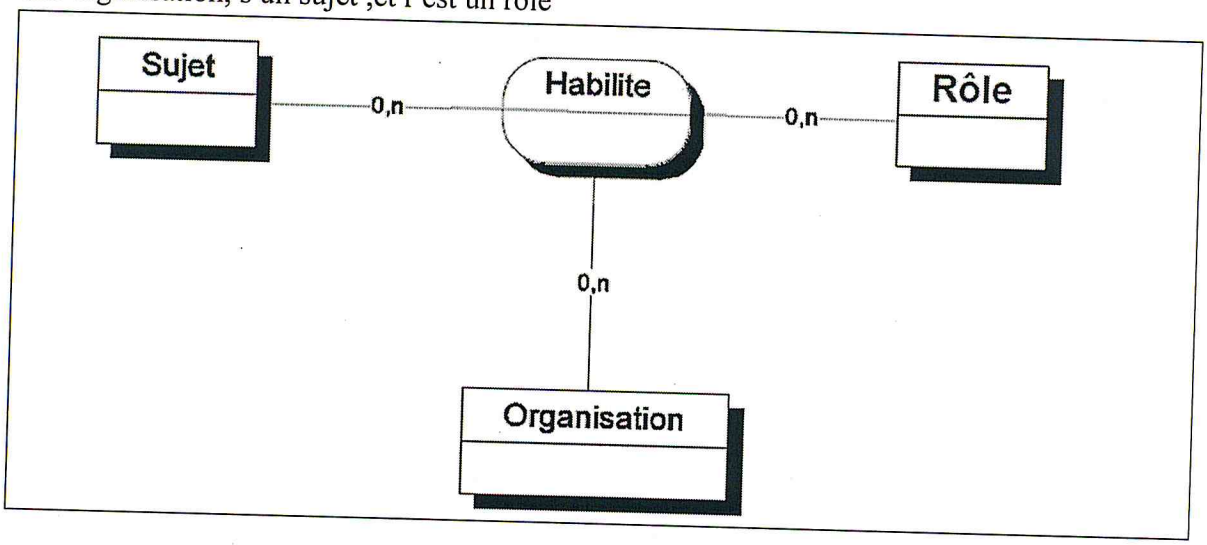

Figure 1.2 : la relation Habilite. **[1]**

Contrairement aux modèles TMAC et RBAC qui ne considèrent que des relations binaires entre les organisations et les sujets ou entre les sujets et les rôles, Le modèle OrBAC définit une relation ternaire entre les organisations, les sujets et les rôles. **[1]**

Les deux exemples suivants illustrent le fait que les sujets sont soit des utilisateurs, soit des organisations :

• Habilite(Frantz Fanon,Lilia, cardiologue) : dans l'organisation Frantz Fanon Lylia est habilitée à être cardiologue .

• Habilite(Frantz Fanon, ICU31, unité_des_soins_intensifs) : " dans l'organisation Frantz Fanon l'unité ICU31 est habilité à être une unité des soins intensifs " .

2.1.3 .Les objets et les vues :

Dans le modèle OrBAC , l' entité Objet représente principalement les entités non actives comme les fichiers, les courriers électroniques, les formulaires imprimés, etc. **[1]**

Dans le domaine médical, nous aurons ainsi à considérer des objets comme les dossiers administratifs, les dossiers médicaux et les dossiers chirurgicaux des patients.

Les rôles permettent de structurer les sujets et de faciliter la mise à jour de la politique de sécurité quand un nouvel utilisateur est ajouté. Dans la mesure où il est également nécessaire de structurer les objets et d' ajouter de nouveaux objets au système, une entité comparable au rôle pour les sujets est nécessaire pour les objets. Elle a été appelée : entité Vue. De manière intuitive, une vue correspond, comme dans les bases de données relationnelles, à un ensemble d' objets qui satisfait une propriété commune. **[1]**

Par exemple dans un système de fichier administratif, la vue " dossiers administratifs" correspond à l' ensemble des dossiers administratifs des patients, alors que la vue " dossiers médicaux" correspond aux dossiers médicaux des patients.

Dans la mesure où les vues caractérisent la manière dont les objets sont utilisés dans l' organisation, une relation qui lie ces trois entités a été créé : la relation Utilise (**Figure 1.3**). Utilise(org, o, v) signifie que org utilise l' objet o dans la vue v Si org est une organisation, o est un objet et v est une vue. **[1]**

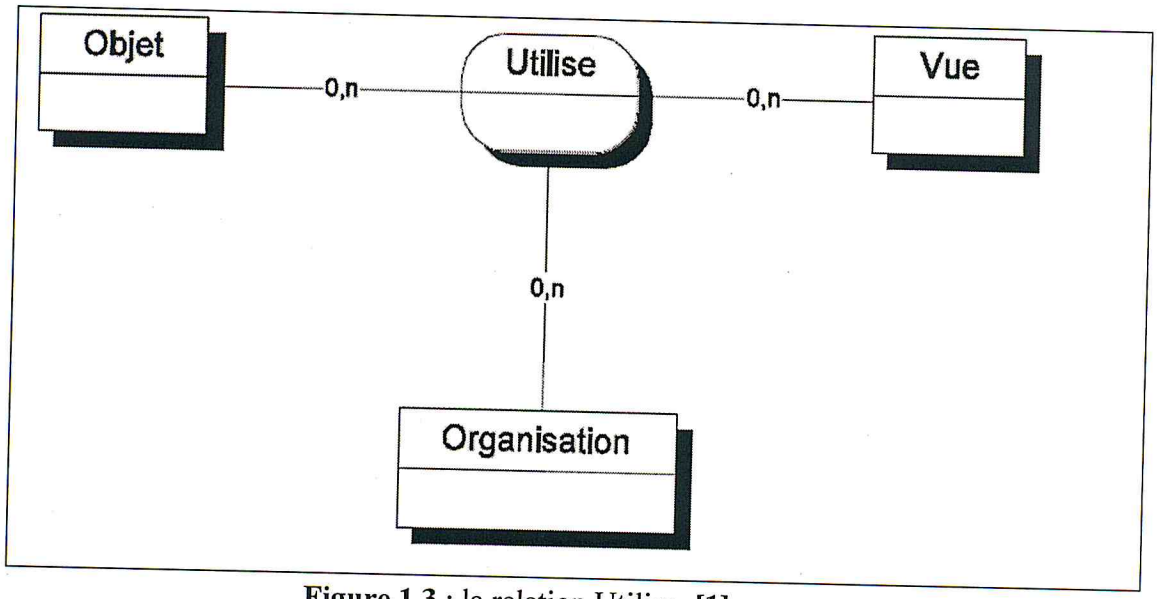

Figure 1.3 : la relation Utilise. **[1]**

Le modèle OrBAC définit donc une nouvelle relation ternaire entre les organisations, les objets et les vues. Ainsi une même vue peut être définie différemment suivant l' organisation considérée. Le but est de caractériser des organisations qui donnent des définitions différentes à une même vue. **[1]**
La vue " dossier médical" peut être définie à l'hôpital Frantz Fanon comme un ensemble de documents Word, et comme un ensemble de documents Latex à l'hôpital Civil :

- Utilise(Frantz Fanon, F31.doc, dossier_médical) : " L' hôpital Frantz Fanon utilise F31.doc comme un dossier médical"
- Utilise(Civil, F32.tex, dossier_médical) : " L' hôpital Civil utilise F32.tex comme un dossier médical" .

2.1.4 .Les actions et les activités :

Les politiques de sécurité spécifient les accès autorisés aux entités passives par des entités actives et régulent les actions opérées sur le système.

Dans le modèle OrBAC, l'entité Action englobe principalement les actions informatiques comme " lire" , " écrire" , " envoyer" , etc. De la même manière que dans les sections **2.1.2 et 2.1.3** où les rôles et les vues sont des abstractions des sujets et des objets, une entité est utilisée comme abstraction des actions : l' entité Activité. Ainsi, les rôles associent des sujets qui remplissent les mêmes fonctions, les vues regroupent des objets qui satisfont une propriété commune et par analogie les activités

correspondent à des actions qui ont un objectif commun. Dans le modèle OrBAC, les activités pourront être " consulter" , "modifier" , " transmettre" , etc. Dans la mesure où des organisations différentes peuvent considérer qu' une même action est employée à la réalisation d' activités différentes, la relation Considère (**Figure1. 4**) sera utilisée pour associer les entités Organisation, Action et Activité. Plus précisément, Considère(org, α, a) signifie que l' organisation org considère l'action α comme faisant partie de l' activité a si org est une organisation, α est une action et a est une activité. **[1]**

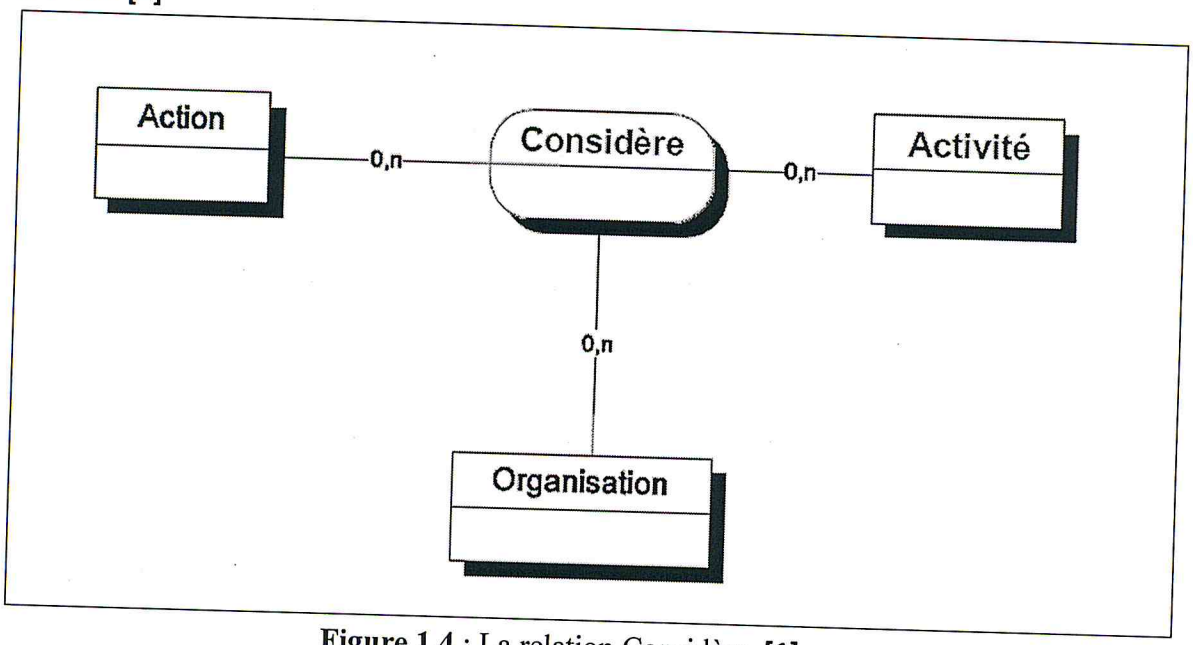

Figure 1.4 : La relation Considère. **[1]**

Remarquons que là encore une relation ternaire est définie. L' objectif est de pouvoir caractériser des organisations qui structurent différemment les mêmes activités. **[1]**

Si nous considérons l'activité " consultation" . Cette activité peut correspondre, dans l' organisation hôpital Frantz Fanon, à l' action " lire" un fichier, mais peut tout aussi bien correspondre à l' action " sélectionner" sur une base de données dans l' hôpital Civil .

• Considère (Frantz Fanon, lire, consultation) : " dans l' hôpital Frantz Fanon lire est considérée comme une consultation" et

• Considère (Civil, sélectionner, consultation) : " dans l' hôpital Civil sélectionner est considérée comme une consultation" .

2.2 . Notion de Contexte:

Les contextes sont utilisés pour spécifier les circonstances concrètes dans lesquelles les organisations accordent des permissions de réaliser des activités sur des vues. Dans le domaine médical, une nouvelle entité Contexte permettra d' exprimer des circonstances telles que " urgence" , "maladie fatale, etc. Les contextes peuvent être vus comme des relations ternaires entre les sujets, les objets et les actions définis dans une certaine organisation. Par conséquent, les entités Organisation, Sujet, Objet, Action et Contexte sont liées par une relation appelée Définit **(Figure 1.5)** tel que : Définit(org, s, α, o, c) signifie qu' au sein de l' organisation org, le contexte c est vraie entre le sujet s, l' objet o et l' action α si org est une organisation, s est un sujet, α est une action, o est un objet et c est un contexte. **[1]**

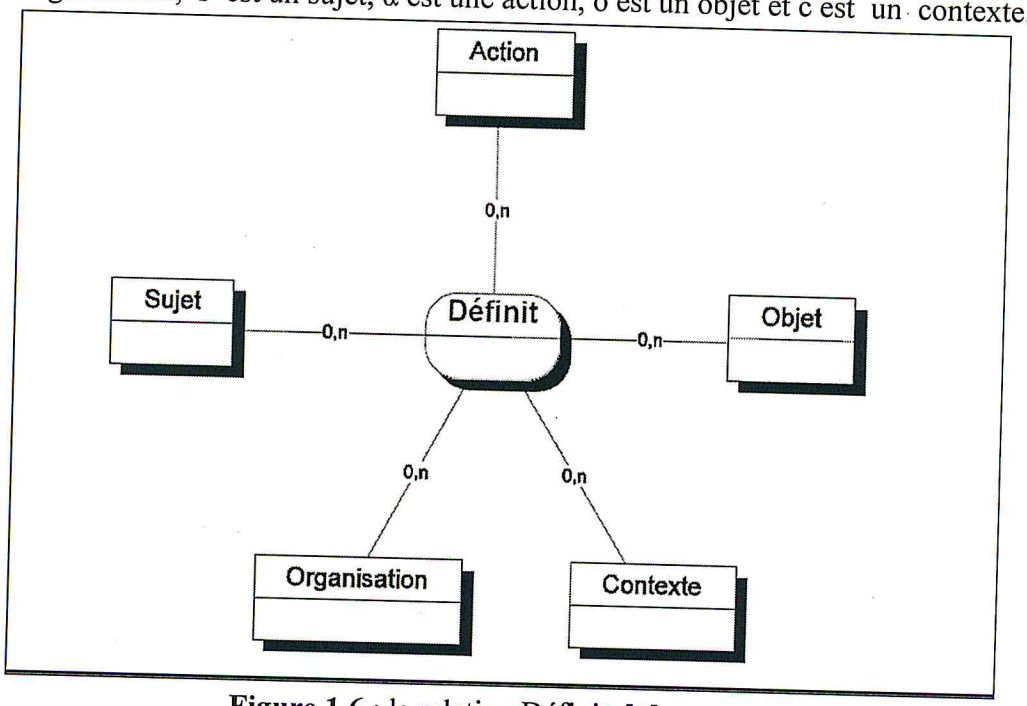

Figure 1.6 : la relation Définit. **[1]**

Considérons les faits suivants :

- Définit (Frantz Fanon, Lylia, lire, F31.doc, urgence) et
- Définit (Civil, Bouchra, lire, F32.tex, médecin_traitant).

Si le premier fait est vrai, alors Lylia n' a pas besoin d' être le médecin traitant du patient correspondant au dossier médical F31.doc pour consulter son dossier. En effet, il est raisonnable de considérer que dans un contexte d' urgence, les médecins ont un accès immédiat à tous les dossiers médicaux. Si le second fait est vrai, alors Bouchra doit être le médecin traitant du patient dont le dossier

médical est F32.tex : dans un contexte normal comme "médecin traitant" , les médecins ont uniquement l' autorisation de consulter les dossiers médicaux de leurs patients.

2.3 .Politique de sécurité :

En utilisant les entités et les relations introduites dans les sections précédentes, des politiques de sécurité appliquées à telle ou telle organisation peuvent être définies .

Une politique de sécurité réglemente les accès au système à travers des permissions, des interdictions, des obligations et des recommandations. **[1]**

Nous ne traiterons que les **permissions**, en considérant que les mêmes raisonnements s'appliquent aux interdictions, aux obligations et aux recommandations.

L'objectif est ici d'ajouter une nouvelle entité Permission afin de relier entre eux les organisations, les rôles, les vues, les activités et les contextes. Plus précisément, si org est une organisation, r est un rôle, a est une activité, v est une vue et c et un contexte, alors Permission (org, r, a, v, c) signifie que dans l'organisation org une permission est accordée au rôle r de réaliser l'activité a sur la vue v dans un contexte c. **[1]**

Prenons l'exemple de l'hôpital ''Frantz Fanon'' qui accorde au rôle "médecin" la permission de réaliser l'activité " consulter" sur la vue " dossier médical" dans le contexte ''urgence''. Cette règle de sécurité est exprimée comme suit :

- Permission (Frantz Fanon, médecin, consulter, dossier_médical, urgence).

2.4. Les Permissions concrètes

La relation Permission, décrite précédemment, permet à une organisation donnée de spécifier les permissions accordées suivant le contexte. De telles permissions correspondent à une relation entre les rôles, les vues et les activités. Pour autant, le contrôle d'accès bas niveau doit permettre de décrire les actions concrètes que réalisent les sujets sur les objets.

Dans le but de modéliser des permissions concrètes, OrBAC introduit la relation Est_permis entre les sujets, les objets et les actions : si s est un sujet, a est une action et o est un objet, alors Est_permis(s, a, o) signifie que le sujet s a la permission de réaliser l'action a sur l'objet o. Dans le modèle OrBAC, les triplets, qui sont des instances de la

relation Est_permis, sont dérivés logiquement des permissions accordées aux rôles, aux vues et aux activités par la relation Permission. **[1]**

3. Interactions du modèle OrBAC

La (**Figure 1.6**) illustre les interactions entre les différents éléments de ce modèle. Il contient sept entités : Organisation, Rôle, Vue, Activité, Sujet, Objet et Action, et six relations : habilite, considère, utilise, définit, Est_permis, permission. **[1]**

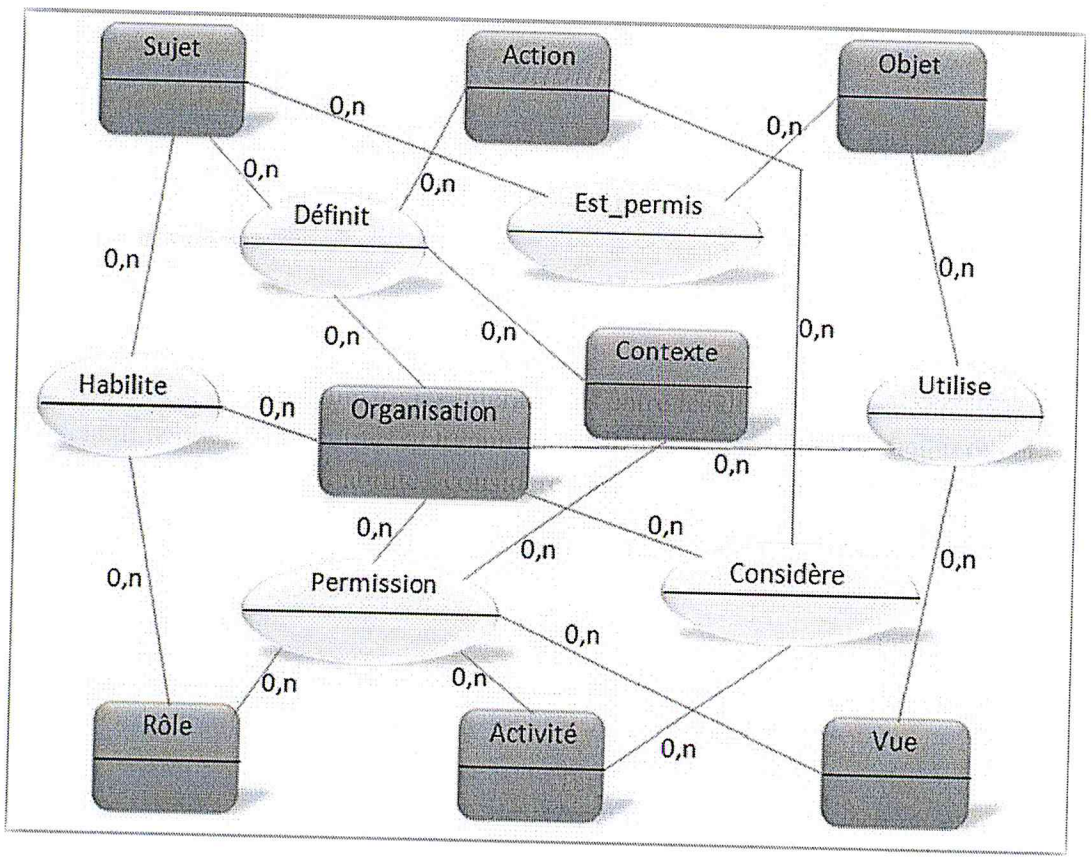

Figure 1.6 : Les interactions du modèle OrBAC.

4. Les hiérarchies dans une organisation :

Dans le modèle Or-BAC, il est possible de définir des hiérarchies de rôles mais aussi des hiérarchies de vues et d'activités. Chaque hiérarchie définit respectivement une relation d'ordre partiel sur l'ensemble des rôles, des vues et des activités. **[2]**

Afin de modéliser par la suite ces différentes hiérarchies, nous présentons dans cette section les règles générales d'héritage des permissions qui leur sont associées.

4.1. Hiérarchie de rôles

Dans un premier temps on s'intéresse à étudier la hiérarchie de rôles. Pour cela on introduit le prédicat sub_role (org, r1, r2) qui signifie : dans l'organisation org, le rôle r1 est un sous-rôle du rôle r2.On remarque que la hiérarchie de rôles dépend de l'organisation. Ainsi, chaque organisation peut définir sa propre hiérarchie de rôles. **[2]**

4.2. Hiérarchie d'activités

Dans cette section on s'intéresse à l'héritage entre les activités. Dans chaque organisation, les activités sont structurées sous forme de hiérarchies. La modélisation de ce type de relation hiérarchique est faite au moyen du prédicat sub_activité (org, a1, a2) qui signifie : dans l'organisation org, l'activité a1 est une sous-activité de a2. **[2]**

La sémantique attribuée à cette hiérarchisation est la spécialisation. Ainsi, dans l'organisation hôpital "Frantz-Fanon", l'activité gestion (des dossiers médicaux par exemple) est spécialisée en trois activités création, consultation et mise-à-jour. La structure hiérarchique associée à l'activité gestion est exprimée au moyen du prédicat sub_activité défini ci-dessus :

sub_activité (Frantz-Fanon, création, gestion),

sub_activité (Frantz-Fanon, consultation, gestion) et

sub_activity(Frantz-Fanon, mise-à-jour, gestion).

4.3. Hiérarchie de vues :

Comme pour les rôles et les activités, l'ensemble des vues est structuré par des hiérarchies dépendant de l'organisation. Cette hiérarchisation est modélisée par le prédicat sub_vue (org, v1,v2) qui signifie : dans l'organisation org, la vue v1 est une sous-vue de la vue v2.

La sémantique attribuée à cette relation hiérarchique est la spécialisation. Cette structuration est en fait proche de la hiérarchie d'héritage de classes utilisée dans les approches orientées objet (la relation Is a). **[2]**

5. Conclusion :

Dans ce chapitre, nous avons présenté le modèle de contrôle d'accès dynamique OrBAC dont l'objectif est de palier aux limites des modèles de contrôle d'accès existants. OrBAC est lié à une notion nommée la délégation, bien que très utilisée, cette notion est très peu modélisée dans les modèles de contrôle d'accès car elle est très complexe, ce concept fera l'objet de notre étude dans le chapitre suivant.

Chapitre 02

Etude des Délégations dans le modèle OrBAC.

Chapitre 02

1. Introduction :

Dans ce chapitre , nous allons introduire en premier lieu le modèle AdOrBac qui est le modèle d'administration de la politique de sécurité OrBac présenté dans le chapitre précédent , nous introduisons ce modèle afin d'introduire en deuxième lieu la notion de délégation qui est spécifiée dans l'un des sous modèles d'AdOrBac et de la détailler plus en exposant une définition plus générale ,ainsi que de présenter ces types en les illustrant avec des exemples concrets .

2. Le modèle AdOrBac :

2.1. Objectifs :

L'administration est un élément important d'une politique de sécurité. Un modèle d'administration doit permettre de déterminer les utilisateurs qui ont le droit de faire évoluer la politique de sécurité.

Sans procédure administrative, on ne pourrait pas contrôler les évolutions de la politique de sécurité.

L'objectif de l'administration de la politique de sécurité est de déterminer par exemple quels rôles sont autorisés à créer de nouvelles permissions, quels rôles sont autorisés à habiliter des sujets dans des rôles…

Le modèle AdOr-BAC permet d'administrer une politique de sécurité Or-BAC. Il permet de gérer toutes ces relations. En particulier, il comporte les trois sous-modèles suivants:

- PRA (Permission-Role Assignment) : Création et suppression de permissions.
- URA (User-Role Assignment) : Habilitation des sujets dans des rôles.
- UPA (User-Permission Assignment) : Affectation de permissions à des utilisateurs(délégation).**[6]**

Pour administrer la politique de sécurité il faut définir des fonctions administratives en considérant différentes vues administratives. Les objets qui appartiennent à ces vues ont une sémantique spéciale. On doit prendre en considération deux vues administratives, la première est la vue **rôle-assignment** et la seconde est **la vue licence assignment**.

La vue rôle-assignment : l'objectif de cette vue est d'assigner à un utilisateur un rôle.

La vue licence assignment: l'objectif de cette vue est de donner la permission à un utilisateur ou un rôle de faire une action ou une activité. **[5]**

La définition des fonctions administratives AdOrbac correspond à la spécification de quel rôle est autorisé d'avoir un accès à des vues administratives grâce à une licence valide.

AdOrbac a une approche homogène avec le modèle Orbac. La syntaxe utilisée pour définir la permission dans la politique administrative est la même que celle définie dans Orbac. **[5]**

Dans Adorbac Les deux vue administratives "licence" et "role-assignment " sont définies comme suit :

2.1.1.La vue Licence :

La vue Licence est utilisée pour spécifier et gérer la politique de sécurité **[5]**, les objets ajoutés à cette vue ont les attributs suivants :

– Bénéficiaire : c'est le sujet bénéficiaire de la licence.
– Privilège : c'est l'action permise par la licence.
– Cible : c'est l'objet auquel la licence accorde l'accès.
– Contexte : c'est les conditions spécifiques qui doivent être satisfaites pour utiliser la licence.

L'existence d'une licence valide est interprétée comme une permission par la règle suivante :

- **Règle Vue Licence**

Permission (Sub, Act, Obj, Context) : – Utilise(L, Licence),Bénéficiaire(L, Suj),Privilège(L, Act),Cible(L,Obj),Contexte(L,Contexte).

Informellement, si (L) appartient à la vue licence :

- (Suj) est le bénéficiaire de (L),
- (Act) est l'action permise par (L),
- (obj) est l'objet auquel la licence accorde l'accès ,
- (Contexte) est la condition spécifique satisfaite pour l'utilisation de (L)

alors le sujet (Suj) a la permission d'effectuer l'action (Act) sur l'objet (Obj) dans le contexte (contexte).**[5]**

2.1.2. La vue Role-assignement :

La vue Role-assignement est associée aux attributs suivants :

– assignee : sujet auquel le rôle est assigné.
– Assignment : le rôle assigné par le rôle-assignment.

Les objets de la vue rôle-assignment sont interprétés par la règle suivante :

- **Règle Rôle- assignment**

Habilite(Sujet, Rôle) : – Utilise(RA,Rôle_Assignment), Assignee(RA, Sujet), Assignment(RA,Rôle).

Informellement, si (RA) appartient à la vue rôle_assignement :

- (RA) est assignée au Sujet,

- (Rôle) est le rôle assigné par le rôle-assignement (RA) alors le Sujet est habilité à faire le Rôle.**[5]**

2.2.La délégation dans AdOrBac :

La vue UPA permet la mise en Œuvre des mécanismes de délégation. Cette tâche n'est pas aisée.

En effet, tous les modèles basés sur les rôles souffrent de l'impossibilité d' affecter des permissions directement aux utilisateurs. Dans de tels modèles, Or-BAC en fait partie, les utilisateurs obtiennent des per missions en fonction du ou des rôles qu'ils jouent. Il est donc impossible d'exprimer par exemple que le rôle « administrateur » peut déléguer, quand il est absent, une partie de ses prérogatives à un utilisateur particulier, et non à un rôle. S'il délègue une permission au rôle chef de département par exemple, tous les utilisateurs jouant ce rôle recevront une permission, ce qui n'est pas souhaitable. UPA permet ainsi à un rôle d'accorder des autorisations à un sujet en particulier, et non à un rôle. **[6]**

3. Définition générale de la délégation:

La délégation permet de donner à un utilisateur particulier un privilège, sans donner ce privilège à toutes les personnes ayant le même rôle que lui. La délégation, bien que très utilisée, est très peu modélisée dans les politiques de sécurité car ce concept est très complexe.

En effet, grâce à une délégation, une permission peut être donnée par le détenteur d'un droit à un tiers pour agir à sa place ou à la place d'un autre.

- On voit déjà ici apparaître qu'une délégation peut faire intervenir plusieurs parties :

- Le sujet qui possède le privilège.
- Le sujet à qui on délègue le privilège.
- Le sujet qui délègue le privilège (pour agir à sa place ou à la place d'un autre).
    - Il existe trois types de de situations dans lesquelles la délégation peut être appliquée :

- la maintenance d'un rôle.
- la décentralisation de l'autorité.
- le travail de collaboration.

La maintenance d'un rôle correspond au cas où un utilisateur doit déléguer une partie de ses permissions afin qu'on puisse remplir toutes ses obligations pendant son absence.

La décentralisation de l'autorité est surtout utile dans le cas où on modifie une partie de l'organisation. En pratique, ce cas peut correspondre à l'ouverture d'un nouvel hôpital

dans lequel on va transférer une partie des médecins exerçant dans les autres hôpitaux de la région.
Le cas du travail en collaboration est évident, si on souhaite que notre partenaire puisse lire les documents que l'on possède sur un projet donné, il faut lui en donner l'autorisation. **[3]**

Remarque :
La délégation pose de nombreux problèmes. Entre autre, un utilisateur X ayant obtenu tous les droits d'un autre utilisateur Y peut ôter les droits à Y si X possède certains droits administratifs. Il peut aussi arriver que l'on oublie de révoquer une délégation faite à Z et qui n'a plus d'utilité d'être, ce qui peut laisser la possibilité à Z de se faire passer pour quelqu'un d'autre. C'est l'une des raisons pour lesquelles il est important de définir deux types de permission, celles qui peuvent être déléguées et celles qui ne peuvent l'être.

4. Différents types de la délégation :

De plus, la délégation peut avoir plusieurs types **(Figure 2.1)**:

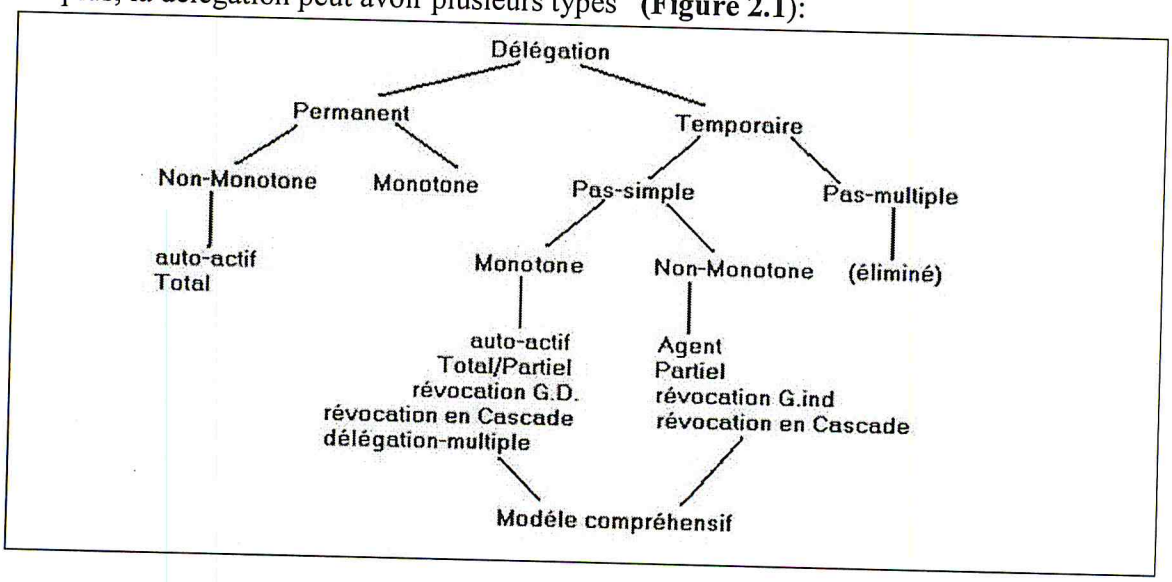

Figure 2.1 : arbre récapitulant les types de délégation et leur possible imbrication.**[4]**

4.1. Délégation temporaire/permanente :
Il faut tout d'abord distinguer les délégations temporaires ou permanentes. En effet, on peut souhaiter qu'un utilisateur ait de façon permanente un droit, afin de ne pas à avoir à renouveler sans cesse ce droit. **[3]**

Exemple :
Prenons l'exemple d'un chef de service d'hôpital qui part en vacances , il délègue temporairement ses droits à l'un de ses chefs d'unités pour la durée de ses vacances et ces

droits seront révoqués automatiquement après la fin de ces vacances là, ce cas est une *délégation temporaire* .Si on reprends l'exemple du chef de service mais cette fois ci il part en retraire et délègue ses droits à un chef d'unité d'une façon permanente cela voudrait dire que cette délégation aura effet jusqu'à nouvel ordre , ce cas-là est une *délégation permanente*. Nous allons suivre cet exemple pour les sections suivantes

4.2. Délégation monotone/non-monotone :

C'est le droit de conserver son privilège quand on le délègue. Dans le cas où la délégation est monotone, la personne qui délègue le droit conserve ce droit. Tandis que si la délégation est non-monotone, la personne qui délègue un droit perd ce droit. **[3]**

Exemple :

si le chef de service souhaite conserver son droit par exemple pendant la durée des vacances en le délégant au chef d'unité ce sera une *délégation monotone* alors que si en délégant ce droit il le perds pour une durée ou définitivement ce sera une *délégation non monotone*.

4.3. Délégation "grant-dependant"/"grant-independant" :

Dans le cas, où la délégation est temporaire et monotone, alors il faut alors choisir quelles sont les personnes susceptibles de révoquer les droits sous-délégués. Si on autorise uniquement la personne ayant déléguée originellement le droit à révoquer un droit délégué, alors on dit que c'est une délégation de type "grant-dependant". Si on autorise toute personne X ayant délégué un droit, avant que Y ait reçu ce droit par délégation, à révoquer ce droit à Y, alors on dit que la délégation est de type "grant-independant". Ce dernier type de délégation permet d'ôter rapidement un droit à une personne qui peut être malveillante, sans avoir à retrouver qui était à l'origine de la délégation (ce qui peut être très fastidieux si l'organisation est importante). **[3]**

Exemple :

Supposons que le chef de service délègue ses droits et qu'il les garde (délégation monotone) et qu'il les délègue pour la durée des vacances (délégation temporaire) , si c'est seulement le chef de service qui a le droit de révoquer la délégation qu'il a faite(à savoir au chef d'unité) ou la délégation qui a été faite par la personne à qui il a délégué on sera dans le cas de délégation « *grant-dependant* ».alors que si le chef d'unité délègue à son tour ce droit à un membre de l'unité et qu'il a le droit de révoquer la délégation qu'il a faite à ce membre on sera dans le cas de délégation "*grant-independant*".

4.4. Délégation totale/partielle :

On peut choisir de déléguer partiellement ou totalement un ensemble de droits. Lorsque l'on souhaite déléguer la totalité de ses droits à quelqu'un, par exemple son remplaçant, alors on applique une délégation totale. Par contre, si on ne veut donner qu'une partie de

ces droits, pour déléguer juste une tâche à quelqu'un, alors on a une délégation partielle. **[3]**

Exemple :
Si le chef de service veut déléguer la totalité de ses droits au chef d'unité pendant son absence , pour que celui-ci soit un remplaçant à part entière on sera dans le cas de délégation totale (une délégation de rôle) , si par contre le chef de service veut déléguer que ses fonctions principales en laissant de côté par exemple les vues sur le courrier ou la suppression de dossier on sera dans le cas d'une délégation partielle (délégation de licence).

4.5. Délégation par agent/auto-active :
Il y a deux façons d'administrer la délégation, si une personne X veut déléguer un droit à Y. La première solution est que c'est X qui administre la délégation. C'est la délégation auto-active. La deuxième solution est que X demande à un agent d'affecter ce droit à Y. C'est la délégation par agent. L'agent pouvant être n'importe quelle tierce personne dans l'organisation. Généralement, l'agent ne peut pas s'affecter les droits qu'il gère. Il est possible de définir le nombre de sous-délégation possible. **[3]**
Exemple :
Si le chef de service pour déléguer ses droits au chef d'unité n'a besoin de passer par aucune tierce personne dans ce cas on aura une *délégation auto-active* c'est-à-dire que le chef de service administre lui-même la délégation, si par contre il est obligé de passé par un intermédiaire par exemple l'administrateur de la politique pour que celui délègue le droit que possède le chef de service au chef d'unité on sera dans le cas de délégation par agent .

4.6. Délégation à un-pas/à pas-multiple :
Dans la délégation à n-pas, un même droit pourra être délégué à une chaîne de n personnes. **[3]**
Exemple :
Par exemple, le chef de service délègue au chef d'unité un droit D (tous les droits ou une partie) à 2-pas et le chef d'unité délègue D à 1-pas à un des membres de son unité dans ce cas la n =2.

4.7. Délégation simple/multiple :
De plus, il faut choisir si lorsqu'on autorise une personne à déléguer, elle peut elle-même déléguer son droit à une unique personne (délégation simple), ou à plusieurs personnes (c'est la délégation multiple). **[3]**
Exemple :
Si le chef de service délègue ses droits à tous les chefs d'unités quel que soit leur nombre (le nombre maximum de bénéficiaires devra être contrôlé) on sera en présence de

délégation multiple si par contre il délègue a un seul chef d'unité à la fois on sera en présence de *délégation simple* .

4.8. Délégation par accord unilatéral/bilatéral :
Pour déléguer un droit, il faut qu'au moins une personne donne son accord. On peut envisager deux types d'accord pour la délégation. L'accord unilatéral ne prend en compte que la décision de la personne désirant déléguer son droit. L'accord bilatéral vérifie que les deux parties, celle qui délègue et celle qui reçoit, sont d'accord. **[3]**

Exemple :
Si pour que le chef de service délègue ses droits au chef d'unité il lui faut l'accord préalable du chef d'unité on est en présence de *délégation bilatérale* alors que si le chef de service peut déléguer ses droits sans se soucier de l'accord du chef d'unité on est en présence de *délégation unilatérale.*

4.9. Révocation de la délégation simple/en cascade :
Si la délégation est temporaire, il faut pouvoir la révoquer. On a pu voir précédemment deux types de délégation jouant sur la révocation. Lorsque la délégation est "grant-dependant" alors seule la personne à l'origine de la délégation peut ôter ce droit. Quand la délégation est de type "grant-independant" seules les personnes ayant engendré la délégation d'un droit à une personne peuvent lui révoquer ce droit. Cependant, la personne, dont les droits ont été révoqués, a peut-être pu déléguer ce droit auparavant. Cette situation peut poser des problèmes dans certains cas. Selon le type de délégation, la personne ayant était déchue d'un droit peut récupérer ce droit grâce à une personne à qui elle aurait délégué le droit. Pour anticiper ce problème, on peut créer deux types de révocation. Un premier type permet de ne révoquer le droit qu'à une personne désignée c'est une révocation simple. Le deuxième type permet de révoquer le droit sur une personne, ainsi que sur toutes les personnes ayant reçu ce droit par délégation, c'est une révocation en cascade. **[3]**

Exemple :
Dans le cas de révocation « grant-dependant » et « grant-indepedant »cela a été expliqué dans la section **4.3** du chapitre , pour ce qui est de la révocation simple ou en cascade supposons que le chef de service ait fait une délégation a 2 pas (comme dans l'exemple de la section **4.6**) si le chef de service en révoquant les droits données par la délégation il révoque que les droits du chef d'unité on est en présence de *révocation simple* , si par contre en révoquant les droits donnés par cette délégation il révoque les droits du chef d'unité ainsi que du membre d'unité on est en présence de *révocation en cascade*.

Remarque :

La délégation a de multiples avantages et offre de nombreuses possibilités. Cependant, il apparaît que si on l'autorise à mauvais escient, alors elle peut aller à l'encontre de la politique. En effet, la révocation d'une délégation permet à une personne d'ôter un droit D à une personne X. Mais que se passe-t-il lorsque la personne X possède ce droit avant la délégation? Des personnes mal intentionnées pourraient ainsi utiliser la délégation afin d'effectuer des révocations qu'ils n'avaient pas le droit de faire. D'où l'importance de bien administrer sa politique de contrôle d'accès, si la délégation est mise en place.

5.Conclusion :

Nous avons présenté dans les deux chapitres précédents le modèle OrBAC et son modèle d'administration afin d'introduire la notion de délégation.

Le travail que nous avons réalisé s'inspire du modèle OrBAC et de son modèle d'administration de délégation. La politique de sécurité est dynamique mais le formalisme utilisé est différent.

OrBAC est basé sur la logique du premier ordre où le contexte est un simple argument. Le modèle sur lequel nous nous sommes basées repose sur la logique de description non monotone.

Avant de présenter les particularités de cette logique, nous présentons dans le chapitre suivant les caractéristiques de base d'une logique de description.

Chapitre 03

Etude des logiques de description.

1. Introduction :

Un système à base de connaissances est un programme capable de raisonner sur un domaine d'application particulier pour résoudre un certain problème grâce aux connaissances relatives au domaine étudié. L'objectif principal des LD consiste à pouvoir raisonner efficacement pour minimiser les temps de réponse. Par conséquent, la communauté scientifique a publié de nombreuses recherches qui portent sur l'étude du rapport expressivité/performance des différentes LD **[14].**

La principale qualité des LD réside dans leurs algorithmes d'inférence dont la complexité est souvent inférieure aux complexités des démonstrateurs de preuves de la logique de premier ordre **[15].**

Ce chapitre présente les logiques de description (LD), une famille de langages de représentation de la connaissance qui peut être utilisée pour représenter la connaissance d'un domaine d'application par un moyen clair, formel et structuré. Ces langages exploitent, en général, des sous-ensembles décidables de la logique de premier ordre. Ils ont été largement étudiés et utilisés dans plusieurs systèmes à base de connaissances.

2. Présentation de la logique de description :

Le nom de logique de description se rapporte, d'une part à la description de concepts utilisés pour décrire un domaine et d'autre part à la sémantique basée sur la logique qui peut être donnée par une transcription en logique des prédicats du premier ordre. La logique de description a été développée comme une extension des frames et des réseaux sémantiques, qui ne possédaient pas de sémantique formelle basée sur la logique.**[28]**

3. Modélisation des connaissances avec la logique des descriptions :

La modélisation des connaissances d'un domaine avec les logiques de description se réalise en deux niveaux (**Figure 3.1**). Le premier, le niveau terminologique ou TBox , décrit les connaissances générales d'un domaine alors que le second, le niveau factuel ou ABox , représente une configuration précise**[8].**

Une TBox comprend la définition des concepts et des rôles, alors qu'une ABox décrit les individus en les nommant et en spécifiant en termes de concepts et de rôles, des assertions qui portent sur ces individus nommés. Plusieurs ABox peuvent être associées à

une même TBox ; chacune représente une configuration constituée d'individus, et utilise les concepts et rôles de la TBox pour l'exprimer. Leur objectif est de lever les ambiguïtés de plusieurs représentations imprécises.

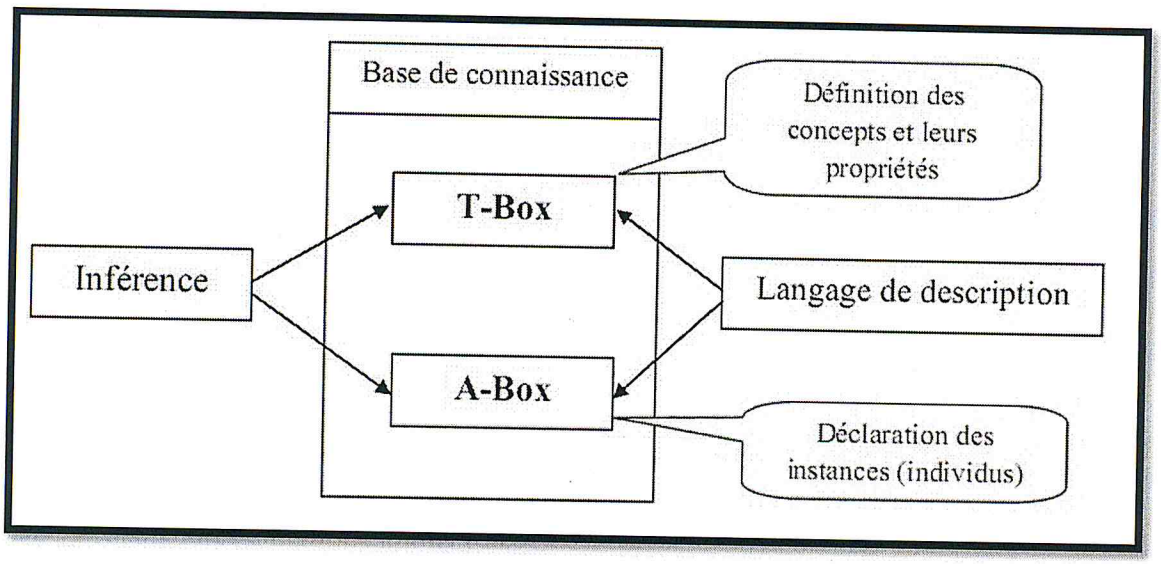

Figure 3.1 : Modélisation de connaissances en deux niveaux.**[24]**

3.1 Niveau terminologique (TBox) :

Le côté gauche du la **Figure 3.2** présente un exemple de TBox. Les prochains paragraphes explicitent divers aspects des TBox en se référant à cet exemple.

TBox	ABox
Femme ≡ Personne ⊓ Femelle Homme ≡ Personne ⊓¬ Femelle Mere ≡ Femme ⊓ ∃aEnfant.Personne Pere ≡ Homme ⊓ ∃aEnfant.Personne Parent ≡Pere ⊔ Mere GrandMere ≡ Mere ⊓ ∃aEnfant.Parent MereAvecPlusieursEnfants ≡Mere ⊓≥3 aEnfant MereSansFille ≡Mere ⊓ ∀aEnfant.¬Femme Epouse ≡ Femme ⊓ ∃aCommeMari.Homme	Femme(Bouchra) Femme(Doria) Pere(Djihad) Homme(Wassim) Mere(Bouchra) aEnfant(Bouchra,Doria) aEnfant (Djihad,Wassim) aEnfant(Bouchra,Wassim)

Figure 3.2 : Une base de connaissances composée d'une TBox et d'une ABox.

- **Les entités atomiques :**

  Les concepts atomiques et rôles atomiques constituent les entités élémentaires d'une TBox. Les noms débutant par une lettre majuscule désignent les concepts, alors que ceux débutant par une lettre minuscule dénomment les rôles (par exemple : les concepts *Femelle* , *Homme* et *Femme*, et le rôle *aEnfant*). **[29]**

La TBox contient toutes les définitions des concepts du domaine ainsi que les rôles. Dans la TBox, on est généralement intéressés à savoir si tous les concepts définis sont consistants, c'est-à-dire si, pour chaque concept, il peut exister au moins un individu membre de cette classe.

Par exemple, si on définit une classe Personne comme étant à la fois une sous-classe des classes Homme et Femme (**Figure3.2**)

Homme ≡ Personne ⊓¬ Femelle
Femme ≡ Personne ⊓ Femelle

Avec une TBox spécifiant que ces deux classes sont disjointes (c'est-à-dire qu'aucune entité ne peut à la fois être un Homme et une Femme),

Par exemple :

Personne(Leila) est une entité.

Femelle(Leila) n'existe pas comme entité.

Alors on ne peut définir dans quelle classe appartient Personne(Leila), On se retrouve avec un concept inconsistant.

3.2. Niveau factuel (ABox) :

La ABox est un niveau factuel (assertion), Dans la ABox, on retrouve les assertions sur les individus. En d'autres mots, on y spécifie quelles sont les entités du monde et à quelle classe elles appartiennent.

- Par exemple : Bouchra est une mère, c'est-à-dire un individu qui est une instance de la classe mère.

La ABox contient aussi des énoncés spécifiant les relations qui existent entre les individus. Ainsi, comme dans la TBox :

- Par exemple : « il sera spécifié qu'une mère doit avoir au moins un enfant », la ABox devra contenir au moins un autre individu « Femme (Doria) », et une relation entre Bouchra et Doria indiquant qu'il est un de ses enfants. aEnfant(Bouchra,Doria) (**Figure3.2**)

Les inférences avec la ABox visent à déterminer si un ensemble d'assertions est consistant, c'est-à-dire si un individu déclaré comme instance d'une classe peut réellement être une instance de cette classe et, similairement, si une relation déclarée entre deux individus est réellement possible.

- **Remarque :**
  On trouve dans la TBox de la **Figure3.2** qu'une mère doit avoir au moins un enfant :
  Mère ≡ Femme ⊓ ∃***aEnfant***.Personne

  Supposons Nous avons dans la Abox:
  Mère(Leila) existe.
  aEnfant(Leila,X) n'existe pas.

  X : l'enfant
  Alors la ABox sera inconsistante, elle contient une mère sans enfant.

3.3. Les composants de base des Logiques de description :

- Les concepts correspondent à des "classes d'éléments" et sont interprétés comme un ensemble dans un univers donné.
- Les rôles correspondent aux "liens entres les éléments" et sont interprétés comme des relations binaires sur un univers donné.
- Les individus correspondent aux éléments d'un univers donné.
- Les opérateurs participent à former des concepts et des rôles qui restreints de manière à ce que :
  - La satisfiabilité et la subsomption soient décidables et si possible, avec une faible Complexité.

3.4. La logique minimale $\mathcal{AL}$:

Pour des fins de simplicité, ce qui suit décrit en premier lieu une logique minimale $\mathcal{AL}$ nommée qui a été introduite par **[16]**.et qui revêt d'une grande importance dans le domaine. Cette logique est minimale, dans le sens où une logique moins expressive représente peu d'intérêt **[17]**.

3.4.1 .Les constructeurs d' $\mathcal{AL}$:

La figure 3.3 illustre les constructeurs offerts par $\mathcal{AL}$ pour l'édification de concepts composés.

$C, D \longrightarrow$	A	(concept atomique)
	$\mid \top$	(le concept universel)
	$\mid \bot$	(le concept le plus spécifique)
	$\mid \neg A$	(la négation atomique)
	$\mid C \sqcap D$	(l'intersection)
	$\mid \exists R.\top$	(quantification existentielle limitée)
	$\mid \forall R.C$	(quantification universelle complète)

Figure 3.3 La grammaire des expressions conceptuelles selon $\mathcal{AL}$ **[17]**.

- Le constructeur (C ⊓ D) permet de faire la conjonction de deux concepts composés, ce qui représente l'ensemble des individus, membres à la fois du concept C et du concept D.
- Le constructeur (¬A) est utilisé pour évoquer la négation d'un concept atomique.
- Le quantificateur existentiel non typé (∃R.T) désigne l'ensemble des individus, membres du domaine d'un rôle R pour une interprétation donnée.
- Le quantificateur universel (∀R.C) évoque l'ensemble des individus du domaine d'un rôle R qui sont en relation, par le biais de R, avec un individu du concept C, pour une interprétation donnée

3.5. Mécanismes d'inférence :

Un système de LD stocke non seulement des terminologies et des assertions, mais offre également les services d'inférence. La charge principale du raisonnement dans une LD est de découvrir des connaissances implicites à partir des connaissances explicites par l'inférence. Les mécanismes d'inférence sont également réalisés sur toutes la TBox ainsi que sur la ABox. **[24]**

Les deux inférences de base dans les LDs sont *la classification* de concepts qui s'effectue au niveau de la TBox, et *la reconnaissance d'instances* qui s'effectue au niveau de la ABox. Ces deux opérations sont basées sur des calculs de relations de subsomption. Cette relation ainsi que les autres opérations terminologiques et assertionnelles seront décrites dans cette section. **[24]**

3.5.1. Subsomption :

Cette relation permet d'ordonner les concepts d'une DL dans une hiérarchie. Nous présentons deux types qui sont la subsomption extensionnelle et structurelle **[16].**

3.5.1.1 .Subsomption extensionnelle :

Un concept C subsume un concept D si et seulement si l'ensemble des instances de C contient l'ensemble des instances de D. Par exemple, l'ensemble des instances du concept Femme est inclus dans l'ensemble des instances du concept Humain **[16]. (Figure3.2)**

3.5.1.2 Subsomption structurelle (ou intentionnelle) :

Un concept C subsume un concept D si et seulement si l'ensemble de ses propriétés est inclus dans l'ensemble des propriétés de D **[16].**

Par exemple, l'ensemble des propriétés du concept Femme contient l'ensemble des propriétés du concept *Etre_Femme* le concept Femme possède toutes les propriétés du concept *Etre_Femme* plus les propriétés qui lui sont spécifiques.

3.5.2 Inférences terminologiques :

Nous présentons dans ce qui suit les principales opérations que l'on trouve au niveau de la TBox qu'on appelle les opérations terminologiques

3.5.2.1 Classification des concepts :

La classification de concepts est l'opération qui permet de placer un concept donnée à la place la plus appropriée dans la hiérarchie. Le processus de classification

permet de découvrir les relations de subsomption qui existent entre un nouveau concept et les concepts présents dans une taxonomie **[24].**

Ce processus s'effectue en Trois phases qui sont :

- Trouver les concepts les plus spécifiques qui subsument le concept D (ce sont les subsumeurs) .
- Rechercher les concepts les plus généraux que D subsume (ce sont les subsumés de D).
- La mise en place des nouveaux liens entre le nouveau concept et les concepts présents dans une taxonomie. **[24]**

3.5.2.2. Héritage :

Un concept hérite des propriétés des concepts qui le subsument. Le mécanisme d'héritage consiste à retrouver toutes les propriétés d'un concept à partir des propriétés des concepts qui le subsument **[24].**

3.5.2.3. Détection d'incohérence :

Cette opération consiste à détecter les concepts incohérents (i.e. qui possèdent une définition incohérente). D'un point de vue extensionnel, un concept incohérent est dénoté par l'ensemble vide (aucun individu n'est instance de ce concept) **[24].**

3.5.2.3 .Complétion :
la complétion permet de retrouver des propriétés hérités du concept et aussi des propriétés déduites logiquement **[24].**

3.5.3 .Inférences assertionnelles :

Les opérations assertionnelles varient d'une DL à une autre. La plupart des ABox comprennent au moins l'opération de reconnaissance d'instances.

3.5.3.1 . Reconnaissance d'instance :

La reconnaissance d'instances consiste à trouver pour un individu donné les concepts les plus spécifiques dont il est instance. L'opération élémentaire utilisée dans cette recherche est l'opération de test d'instance (*instance checking* en anglais) qui consiste à vérifier si un objet o est une instance d'un concept C**[24].**

La méthode abstraction- classification est l'une des méthodes employées pour faire de la reconnaissance d'instances. Elle consiste à calculer l'abstraction d'un objet (i.e. transformer un objet en un concept appelé "concept abstrait") et de classer le concept obtenu. Pour calculer le concept abstrait d'un individu, on retrouve toutes les informations liées à l'individu et on les rassemble sous la forme d'une définition de concept la plus spécifique possible. Pour savoir si un objet o1 est une instance d'un concept C, on calcule le concept abstrait Ao1, on teste ensuite si C subsume Ao1. Si c'est le cas, on déduit alors que o1 est une instance du concept C **[24].**

3.5.3.2 .Propagation :

Une assertion sur un individu o1 peut avoir des conséquences logiques sur des individus en relation avec o1. Les DL qui bénéficient de l'opération de propagation propagent les nouvelles informations déduites sur les individus concernés **[24].**

3.5.3.3. Requêtes :

L'utilisateur d'une base de connaissances peut faire des requêtes en posant par exemple des questions de type : 'Quels individus sont en relation par r avec l'individu I ?' **[24].**

3.5.3.4 Fermeture de rôle :

Certaines LDs possèdent une opération de fermeture de rôle.
Si un rôle r est fermé pour un individu I, alors il n'existe pas d'autres individus en relation, par r, avec I que ceux exprimés explicitement **[24].**

3.5.3.5.Chaînage avant :

Étant donné des règles ayant des concepts en prémisse et en conclusion, si on a trouvé un individu qui satisfait un concept se trouvant en prémisse d'une règle, on déduit qu'il satisfait aussi le concept se trouvant en conclusion de cette règle. Cette déduction est effectuée à l'aide d'un mécanisme fonctionnant en chaînage avant **[24].**

4. Logique de description et logique du premier ordre :

Dans cette section nous faisons la comparaison entre la logique des prédicats comme étant un système de représentation de connaissance et les logiques de description .

Les DLs classiques sont des formalismes qui représentent un sous ensemble de la logique du premier ordre. Les objets des DLs peuvent être vus comme des prédicats dans la logique du premier ordre par exemple, les concepts des DLs sont vus comme des

prédicats unaires dans la logique des prédicats et les rôles comme des prédicats binaires, par contre il n'y a pas l'équivalent des prédicats n-aires (avec n>2) dans les DLs classiques **[11].**

Cependant, certaines connaissances ne peuvent être représentées par des formules de la logique des prédicats. De plus, il existe un type de connaissances pour lequel les DLs sont plus adaptées que la logique du premier ordre : les connaissances nécessitant une structuration (i.e. une organisation taxonomique) et une agrégation (i.e. un regroupement) **[11].**

La comparaison d'un point de vue complexité est simple et se résume en deux points :

- Le calcul des prédicats est semi-décidable.
- Les DLs sont des sous-ensembles de la logique du premier ordre pour lesquels les algorithmes d'inférences sont la plupart du temps décidables et parfois même polynomiaux. **[11]**

5. Défaut et exceptions dans la logique de description :

De nombreux travaux ont été introduits dans le but d'étendre les DLs et d'arriver à un meilleur niveau d'expressivité et aboutir à des algorithmes d'inférence complets et de complexité polynomiale.

Tout d'abord les travaux de J.Quant et V.Royer ont été réalisés dans l'intérêt d'ajouter aux DLs une forme de raisonnement par défaut .Les défauts ne sont pas autorisés dans la définition même du concept mais ils sont exprimés sous la forme de règles incidentes de type C1 → C2 . **[18]**

C1 → C2 veut dire : si un objet est une instance du concept C1 alors c'est aussi une instance du concept C2 à moins d'un conflit avec une autre connaissance. Un conflit est caractérisé par l'appartenance à deux concepts disjoints (l'intersection de leurs propriétés est vide). Si le conflit porte sur des propriétés strictes, l'information posant problème est rejetée. Par contre, si l'objet est instance par défaut de deux concepts disjoints, les auteurs supposent qu'il y a une exception sur un des défauts et un processus de résolution de conflits est lancé **[24].**

Pour l'approche de L. Padgham, B. Nebel**[12]** puis L. Padgham et T. Zhang **[25],** un concept possède une définition scindée en deux parties : une partie (appelée *stricte*) qui comprend les propriétés strictes et une partie (appelée *default*) qui comprend les propriétés défauts. **[24]**

Exemple:

Jeune-enfant = enfant Π age-inf-3 / parler.

Décrit l'ensemble des instances qui sont exactement enfant et âgé de minimum 3 ans et qui ont pour propriété incidente le fait de parler.

L. Padgham et B. Nebel associent au concept jeune-enfant deux nœuds correspondant aux deux parties de la définition :

un nœud jeune-enfantstricte et un nœud jeune -enfantdefault avec :

jeune-enfantstricte = enfant Π age-inf-3

jeune-enfantdefault = enfant Π age-inf-3 Π parler

Les auteurs distinguent deux sortes de subsomption: une subsomption stricte et une subsomption défaut. Le nœud stricte d'un concept subsume toujours son nœud défaut **[24]**.

De nouveaux connecteurs δ et ε ont été introduit par P. Coupey et C. Fouqueré **[20].** Dans le but de représenter des connaissances par défauts et de type exception d'où la naissance de la nouvelle logique de description qui est **ALδε** qui introduit la notion de défaut et exception dans la définition du concept ce que les autres approches ne permettent pas **[24].**

6. Présentation de la logique de description ALδε :

La logique de description ALδε permet d'introduire les notions défauts et exception, qui sont très importantes pour la représentation non monotone de connaissances, symbolisées par les deux connecteurs δ et ε.

6.1 Syntaxe d' ALδε : (Figure 3.4)

La logique ALδε est définie à partir d'un ensemble R de rôles primitifs, d'un ensemble P de concepts primitifs, des constantes $\top$ et $\bot$ et de la règle syntaxique suivante : (C et D sont des concepts).

C, D	→	$\top$	Le concept le plus général
	|	$\bot$	Le concept le plus spécifique
	|	P	Concept primitif
	|	$C \sqcap D$	Conjonction de concepts
	|	$\neg P$	Négation de concept primitif
	|	$\forall r : C$	Restriction de valeurs pour les rôles (R>0)
	|	δC	Concept défaut
	|	C^{ε}	Exception sur un concept

Figure 3.4 : Syntaxe d' ALδε.**[24]**

6.2. Description des connecteurs δ et ε :

Nous allons décrire les connecteurs δ et ε via des exemples. Le connecteur δ représente la notion de *défaut* et le connecteur ε représente une *exception*, par exemple : le concept « Oiseau » est défini comme un animal qui a un bec et des ailes et qui généralement vole.

$$\text{Oiseau} \equiv \text{animal} \sqcap \text{avec-bec} \sqcap \text{avec-ailes} \sqcap \delta \text{ vole}.$$

Les instances du concept Oiseau possèdent nécessairement les propriétés animal, avec-bec et avec-ailes (propriétés strictes), mais peuvent ne pas posséder la propriété vole (propriété défaut).

Malheureusement, en utilisant cette définition du concept Oiseau on peut par exemple inférer qu'un pingouin qui est un animal, qui a un bec et des ailes est un « Oiseau » puisqu'il possède les propriétés strictes animal, avec-bec et avec-ailes.

$$\text{Pingouin} \equiv \text{animal} \sqcap \text{avec-bec} \sqcap \text{avec-ailes}.$$

Ainsi, comme le dit *R. J. Brachman* la classification semble impossible en présence de connaissance par défaut puisqu'une propriété défaut n'est pas, par définition, nécessaire. **[26]**

Pour résoudre ce problème, *P. Coupey et C. Fouqueré* le connecteur ε qui présente une "exception" à un concept, dans le but de résoudre ce problème en exprimant la propriété suivante:

« Un objet est une instance d'un concept C si et seulement s'il satisfait les propriétés strictes de C et satisfait ou est explicitement exceptionnel par rapport aux propriétés défaut de C ». **[20]**

Donc on ne peut plus inférer qu'un pingouin est un oiseau puisqu'il ne possède pas la propriété vole et il n'est pas exceptionnel par rapport au fait de voler (il ne possède pas la propriété vole ε).

Par contre, le concept poule sera classé sous le concept oiseau puisqu'il vérifie les propriétés strictes du concept oiseau (animal, avec-bec, avec-ailes) et il est exceptionnel par rapport au fait de voler (il possède vole ε).

$$\text{Poule} \equiv \text{animal} \prod \text{avec-bec} \prod \text{avec-ailes} \prod \text{vole}\varepsilon$$

6.3 . Propriétés des connecteurs δ et ε :

Voici les axiomes ci-dessous qui représentent les propriétés des connecteurs δ et ε qui sont d'une grande importance dans le raisonnement basé sur une logique qui introduit les notions de défaut et exception :

1. $(\delta A)\varepsilon \equiv A\varepsilon$
2. $\delta(A \prod B) \equiv (\delta A) \prod (\delta B)$
3. $A \prod \delta A \equiv A$
4. $A\varepsilon \prod \delta A \equiv A\varepsilon$
5. $\delta\delta A \equiv \delta A$

On exprime dans l'axiome 1 le fait qu'une exception n'a de sens que si elle porte sur une propriété défaut. **[20]**

Dans l'axiome 2, on représente la propriété de distributivité du connecteur δ par rapport à la conjonction de deux concepts. On définit dans les axiomes 3 et 4 les relations de subsomption qui existent entre A et δA (δA subsume A) et entre Aε et δA (δA subsume Aε). Et l'axiome 5 permet la suppression de chaînes de défauts redondantes **[20].**

6.4 .Caractéristiques d'ALδε :

- **Point de vue héritage :**

Dans un cadre strict (dans le cadre d'une DL ne comportant pas de connecteur défaut), le mécanisme d'héritage est similaire au mécanisme de subsomption : un concept hérite de toutes les propriétés des concepts qui le subsument.

Or, dans le cadre d'ALδε qui comporte des connaissances de type défaut et exception. Un concept ne pourra pas hériter de toutes les propriétés d'un concept qui le subsume **[24].**

Exemple :

Le concept oiseau hérite des propriétés animal, avec-bec, avec-ailes et δ vole.

Le concept Poule est subsumé par le concept oiseau, il hérite des propriétés animal, avec-bec et avec ailes mais il n'hérite pas la propriété vole puisqu'elle est exceptée.

- **Points forts :**

Un des points forts de l'approche de P. Coupey et C. Fouqueré est d'introduire des défaut au sein même de la définition des concepts et non on tant que règles comme c'est le cas dans les autres approches. P. Coupey et C. Fouqueré montrent dans **[20].** que l'intr-oduction des connecteurs δ et ε améliore considérablement les capacités du mécanisme de classificatin. En effet, pouvoir décrire des concepts avec des propriétés par défaut permet de définir complètement des concepts qui ne pourraient l'être en utilisant uniquem ent des propriétés strictes **[24].**

- **Points faibles :**

D'une part, la logique de description ALδε n'est pas utilisable dans un cadre pratique en r aison de sa faible puissance d'expressivité. D'autre part, la logique de description ALδε n'a pas été dotée d'algorithmes d'inférence permettant de faire évoluer la base de connaissances **[24].**

7.Conclusion :

Nous avons présenté dans ce chapitre les logiques de description classiques d'une manière générale et particulièrement non monotones ainsi que les différents types de subsomption qui sont la subsomption extensionnelle et structurelle.

Notre intérêt s'est porté sur la logique ALδε qui a été développé par P. Coupey et C. Fouqueré. Cette logique a des avantages tels que le pouvoir de définir des propriétés défauts au sein même d'une description d'un concept, en revanche elle a l'inconvénient de ne pas être utilisée grandement dans le cadre pratique. Parmi ses faiblesses, la puissance d'expressivité et le manque d'algorithmes qui font évoluer la base de connaissances tels que la subsomption et l'héritage.
Les insuffisances de cette logique ont fait apparaitre une nouvelle logique plus complète appelée C-Classicδε .Nous présenterons cette logique dans le chapitre qui suit.

Chapitre 04

Etude de la logique CClassicδε.

1. Introduction :

Dans ce chapitre nous allons introduire la logique **CClassic$_{\delta\varepsilon}$** qui est une extension des logiques de descriptions, la logique **CClassic$_{\delta\varepsilon}$** comprend tous les connecteurs de la DL C-CLASSIC plus les connecteurs δ et ε d'ALδε définie par **P. Coupey** et C. **Fouqueré [20]** permettant de représenter des connaissances défaut et exception.

Nous commencerons par une brève présentation de la DL C-CLASSIC,ensuite nous présenterons la syntaxe de la logique **CClassicδε** , ainsi que sa sémantique tout en prenant soin de détailler ses types, et enfin nous finirons par la présentation de la forme normale d'une description et de l'héritage comme mécanisme d'inférence dans le cadre de la logique **CClassicδε** .

2. Présentation de la logique de description C-CLASSIC :

La logique de description C-CLASSIC est décrite par W.W.Cohen et H.Hirsh **[27].** C'est une variante des logiques de descriptions classiques elle comprend un algorithme de calcul de subsomption polynomial, correct et complet.

3. Syntaxe de la logique CClassicδε :

La logique CClassicδε comprend l'ensemble des connecteurs de CClassic et les connecteurs défaut (δ) et exception (ε) issues de la logique $\mathcal{AL}$δε.

La syntaxe est définie à partir d' :

- Un ensemble R de rôles primitifs ;
- Un ensemble P de concepts primitifs ;
- Des constantes $\top$ et $\bot$ qui correspondent respectivement au concept le plus général et au concept le plus spécifique ;
- Un ensemble I d'individus ;
- De la règle syntaxique suivante : (sachant que C et D sont des concepts, u est un nombre réel, n est un nombre entier et li des individus).

C⟶ **T**	Concept le plus général
⊥	Concept le plus spécifique
P	Concept primitif
ONE-OF $\{I_1, I_2, I_3...I_N\}$	Concept en extension
MIN $_u$	Ensemble de réel >u
MAX $_u$	Ensemble de réel ≤u
C ⊓ D	Conjonction de deux concepts
∀R:C	Precise de co-domaine de R
R FILLS $\{I_1, I_2, I_3...I_N\}$	Restriction de valeurs pour R
R AT-LEAST n	Restriction de nombre pour R (minimum)
R AT-MOST n	Restriction de nombre pour R (maximum)
δC	Concept C par défaut
C^{ε}	Exception sur le concept

Voici quelques exemples de descriptions dans la logique CClassicδε:

- Séisme-dangereux ≡ séisme ∏∀ durée : MIN 45 ∏δ (∀degré : MIN 7).
  Ce concept regroupe l'ensemble des instances qui sont des séismes de durée supérieure à 45 secondes et qui ont généralement un degré supérieur à 7 à l'échelle de Richter.
- Séisme-faible ≡ séisme ∏ ∀ durée : MAX 15 ∏ (∀ degré : MIN 7) ε.

Ce concept regroupe l'ensemble des instances qui sont des séismes de durée inférieure à 15 secondes et qui sont exceptionnels par rapport au fait d'avoir un degré supérieur à 7 à l'échelle de Richter.

4. Sémantique :

La logique **CClassicδε** est dotée d'une sémantique afin de formaliser les différentes descriptions de concepts et de pouvoir inférer sur ces descriptions.

Nous présentons deux types de sémantique qui sont la sémantique descriptive et la sémantique intensionnelle (appelée aussi structurelle)**[24].**

4.1 .Sémantique descriptive :

Dans une sémantique descriptive, les concepts sont décrits via un système équationnel appelé EQ, qui souligne de manière formelle les principales propriétés des connecteurs de CClassicδε.

4.1.1 .Système équationnel EQ pour $CClassic_{\delta\varepsilon}$: [24]

$\forall$ A, B, C $\in$ $CClassic_{\delta\varepsilon}$, li $\in$ I :

1. (A $\prod$ B) $\prod$ C ≡ A $\prod$ (B $\prod$ C)
2. A $\prod$ B ≡ B $\prod$ A
3. A $\prod$ A ≡ A
4. T $\prod$ A ≡ A
5. $\perp$ $\prod$ A ≡ $\perp$
6. ($\forall$ R: A) $\prod$ ($\forall$ R: B) ≡ ($\forall$ R: A $\prod$ B)
7. $\forall$ R: T ≡ T
8. ONE_OF E1 $\prod$ ONE_OF E2 ≡ ONE_OF (E1 $\prod$ E2)
9. MIN m $\prod$ MIN n ≡ MIN maxi (m, n)
10. MAX m $\prod$ MAX n ≡ MAX mini (m, n)
11. R FILLS E1 $\prod$ R FILLS E2 ≡ R FILLS (E1 $\prod$ E2)
12. R FILLS φ ≡ T
13. R AT_LEAST n $\prod$ R AT_LEAST m ≡ R AT_LEAST maxi (n, m)
14. R AT_LEAST 0 ≡ T
15. R AT_MOST m $\prod$ R AT_MOST n ≡ R AT_MOST mini (n, m)
16. R AT_MOST 0 ≡ $\forall$ R: $\perp$
17. R FILLS {I1, …, In} ≡ R FILLS {I1, …, In} $\prod$ R AT_LEAST n
18. $\forall$ R: ONE_OF {I1, …, In} ≡ $\forall$ R: ONE_OF {I1, …, In} $\prod$ R AT_MOST n
19. R AT_LEAST n $\prod$ $\forall$ R: ONE_OF {I1, …, In} ≡ R AT_LEAST n $\prod$ $\forall$ R: ONE_OF {I1, …, In} $\prod$
20. R FILLS {I1, …, In}
21. R AT_MOST n $\prod$ R FILLS {I1, …, In} ≡R AT_MOST n $\prod$ R FILLS {I1, …, In} $\prod$ $\forall$ R: ONE_O {I1, …, In}
22. (δA)ε ≡ Aε
23. δ(A $\prod$ B) ≡ (δA) $\prod$ (δB)
24. A $\prod$ δA ≡ A
25. Aε $\prod$ δA ≡ Aε
26. δδA ≡ δA

4.1.2 .Subsomption descriptive :

Du point de vue descriptif, le calcul de subsomption consiste à comparer les termes de CClassicδε via le système équationnel présenté précédemment. Un concept C subsume descriptivement un concept D (noté : D ⊆d C), si et seulement si C $\prod$ D ≡EQ D. **[24]**

Voici un exemple qui illustre le calcul de subsomption descriptive dans la logique $CClassic_{\delta\varepsilon}$ En suivant l'exemple donné dans le Chapitre précédent « l'Oiseau et la Poule »:

Oiseau ≡ animal ∏ avec-bec ∏ avec-ailes ∏ δ vole.

Poule ≡ animal ∏ avec-bec ∏ avec-ailes ∏ voleε

Oiseau subsume t-il descriptivement le concept Poule ?

Nous allons appliquer le système équationnel EQ pour répondre à la question.

Oiseau ∏ Poule ≡ animal ∏ avec-bec ∏ avec-ailes ∏ δ vole ∏
animal ∏ avec-bec ∏ avec-ailes ∏ voleε

≡ animal ∏ avec-bec ∏ avec-ailes ∏ δ vole ∏ voleε

(Axiome 2 et 3)

≡ animal ∏ avec-bec ∏ avec-ailes ∏ voleε

(Axiome 24)

≡ Poule

D'où on déduit que le concept Oiseau subsume le concept Poule.

D'un point de vue algorithmique, il n'est pas facile de manipuler les termes dans le cadre de subsomption. Cela a conduit à adopter un point de vue structurel qui permet d'une part de formaliser le calcul de subsomption et d'autre part de doter $CClassic_{\delta\varepsilon}$ d'une sémantique intentionnelle. **[24]**

Remarque : Nous donnons l'algorithme de calcul de subsomption en Annexe.

4.2 .Sémantique intentionnelle :

Dans une sémantique intentionnelle (structurelle), on décrit un concept en donnant l'ensemble de ces propriétés.

Exemple : Etudiant_admis ≡ Etudiant ⊓ ∀moyenne : MIN 10.

4.2.1. Subsomption intentionnelle :

On dit qu'un concept C subsume intentionnellement (ou structurellement) un concept D si et seulement si D satisfait les propriétés strictes de C et satisfait ou il est explicitement exceptionnel par rapport aux propriétés défauts de C.

Exemple : Soit Oiseau et Poule deux concepts tels que :

Oiseau ≡ animal ∏ avec-bec ∏ avec-ailes ∏ δ vole.

Poule ≡ animal ∏ avec-bec ∏ avec-ailes ∏ voleε..

❖ Question: Le concept oiseau subsume t-il intentionnellement le concept poule ?
- Le concept Poule satisfait toutes les propriétés strictes du concept Oiseau (animal, avec-bec et avec-ailes) et il est explicitement exceptionnel par rapport au fait qu'il vole (voleε), donc d'après la définition de la subsomption intentionnelle le concept « Oiseau » subsume le concept « Poule ».

5. Forme normale d'une description CClassicδε :

Dans CClassicδε, il est difficile de manipuler les descriptions de concept dans le cadre algorithmique. En effet pour pouvoir déterminer une relation de subsomption entre deux concepts à partir de leurs descriptions, il faut pouvoir les comparer. Pour cela, on définit une forme normale descriptive qui représente la description d'un concept.

La forme normale d'un concept défini avec la description T (note fn(T)) est un couple **<tσ, tδ>** où **tσ** contient les propriétés strictes de T et **tδ** ses propriétés défauts **[24].**
tσ et tδ sont des sextuplets de la forme **(dom, min, max, π, r, ε) [24]** avec :

- **dom**: est un ensemble d'individus si la description contient une propriété ONE-OF, sinon le symbole UNIV.
- **min** (respectivement **max**): est un réel si la description contient une propriété MIN (respectivement MAX), sinon le symbole MIN-R (respectivement MAX-R).
- **π:** est un ensemble de concepts primitifs appartenant à la description T.
- **r:** est de la forme <R, fillers, least, Most, c> où :
  - R: est un nom de rôle.
  - fillers: est un ensemble d'individus si la description contient une propriété r FILLS, sinon ϕ.
  - least: (respectivement Most) est un entier si la description contient une propriété

AT-LEAST (respectivement AT-MOST), sinon 0 (respectivement NOLIMIT).

- c: est la forme normale de C si la description contient une propriété ∀R: C.

- **ε:** est un ensemble de sextuplets de la forme (dom, min, max, π, r, ε)

Remarque : L'algorithme de normalisation est présenté en annexe.

6. Héritage :

La subsomption est une procédure très importante qui permet d'organiser les concepts et les classifier dans la taxonomie. Un concept qui est subsumé par un autre hérite de ces propriétés ce qui permet une déduction de toutes les propriétés que possède ce concept **[23].**

Dans cette partie nous détaillerons l'héritage dans le cadre de $CClassic_{\delta\varepsilon}$. Nous expliquons la procédure d'héritage via l'exemple qui suit :

Le concept Oiseau hérite des propriétés : animale, avec-bec, avec-ailes et vole
Oiseau ≡ animal ⊓ avec-bec ⊓ avec-ailes ⊓ δvole

Le concept Poule est subsume par le concept Oiseau, il hérite des propriétés animal, avec-bec, avec-ailes mais il n'hérite pas la propriété vole puisqu'elle est exceptée,

Poule ≡ Oiseau ⊓ voleε

Poule ≡ animal ⊓ avec-bec ⊓ avec-ailes⊓ voleε

Remarque : L'algorithme d'héritage est présenté en annexe.

7. Conclusion

Dans cette partie, nous avons d'abord présenté la logique de description $CClassic_{\delta\varepsilon}$ qui supporte les connaissances de type « défaut » et de type « exception » ensuite sa syntaxe et sémantique , nous avons finis par le mécanisme d'inférence d'héritage dans son cadre .

La logique $CClassic_{\delta\varepsilon}$ est une logique non monotone utilisable dans le cadre pratique comme dans le cadre de notre modèle de contrôle d'accès car elle est assez expressive et ses algorithmes d'inférence sont complets, corrects et de complexité polynomiale.

Nous rappelons que le but de ce travail est de construire un modèle de contrôle d'accès dynamique $Delegation\_OrBAC_{\delta\varepsilon}$ qui sera présenté dans les chapitres qui suit.

Chapitre 05

Modèle de contrôle d'accès dynamique

DELEGATION OrBACδε

Chapitre 05

1. Introduction :

Dans ce chapitre, nous allons nous intéresser à la modélisation du modèle de contrôle d'accès dynamique et contextuel DELEGATION-OrBAC δε en logique de description augmentée avec les connecteurs défaut (δ) et exception (ε) .Ces connecteurs nous permettent d'introduire des connaissances défaut et exception nous donnant un bon niveau d'expressivité, de plus ce modèle est inspiré d'OrBAC et comprends la notion de délégation.

2. Représentation de connaissances :

Dans cette section nous allons introduire le passage du modèle entité-relation du modèle OrBAC définit dans le premier chapitre au modèle en logique de description avec défaut et exception.

La transformation de la taxonomie des entités dans la conception entité-relation en

taxonomie DL se fait comme suit :

- Les entités du modèle (organisation, rôle, activité, vue, sujet, action, objet) sont traduites en concepts atomiques et deviennent les entités élémentaires de la TBox .
- Les relations binaires se traduisent en rôles atomiques.
- Pour les relations n-aires (>2), il n'existe pas d'équivalent au niveau de la logique de description.

Ce problème est typique des langages de représentation de connaissance. Il existe une approche appelée *la réification*, qui consiste à créer un concept pour la relation n-aires et une propriété pour chacun des rôles de la relation du modèle entité-relation.

3. Modélisation de DELEGATION-OrBAC δε :

La modélisation de notre modèle de contrôle d'accès dynamique fut à travers l'écriture des axiomes de la TBOX , de la définition de la ABOX , ainsi que de la présentation des différents mécanismes d'inférence modélisés pour les inférences sur la base de connaissances qui comprends cette TBOX et cette ABOX.

Remarque :

Dans notre système, nous considérons que :

- *Par défaut, le contexte est normal,*
- *Toute action qui n'est pas permise est interdite.*

3.1 .Axiomes de la TBox :

Nous définissons dans cette partie les différents axiomes de la TBox de notre base de connaissances.

Remarque :
Etant donné que Les LDs ne permettent pas d'établir des relations de dimension supérieure à 2 nous avons définis dans les axiomes suivants des relations binaires intermédiaires pour définir chaque axiome exemple : HabiliteS, HabiliteR, HabiliteOr dans la relation Habilite.

3.1.1. Axiome d'attribution de rôle :

Nous avons :

Sujet $\sqsubseteq$ T

Rôle $\sqsubseteq$ T

Organisation $\sqsubseteq$ T

Habilite $\sqsubseteq$ HabiliteS.Sujet $\sqcap$ HabiliteR.Role $\sqcap$ HabiliteOr.Organisation

Cet axiome définit la relation entre un sujet, un rôle et une organisation en utilisant les

Relations binaires : HabiliteS, HabiliteR, HabiliteOr tel que :

- HabiliteS : relie le concept Habilite au concept Sujet.
- HabiliteR : relie le concept Habilite au concept Rôle.
- HabiliteOr : relie le concept Habilite au concept Organisation.

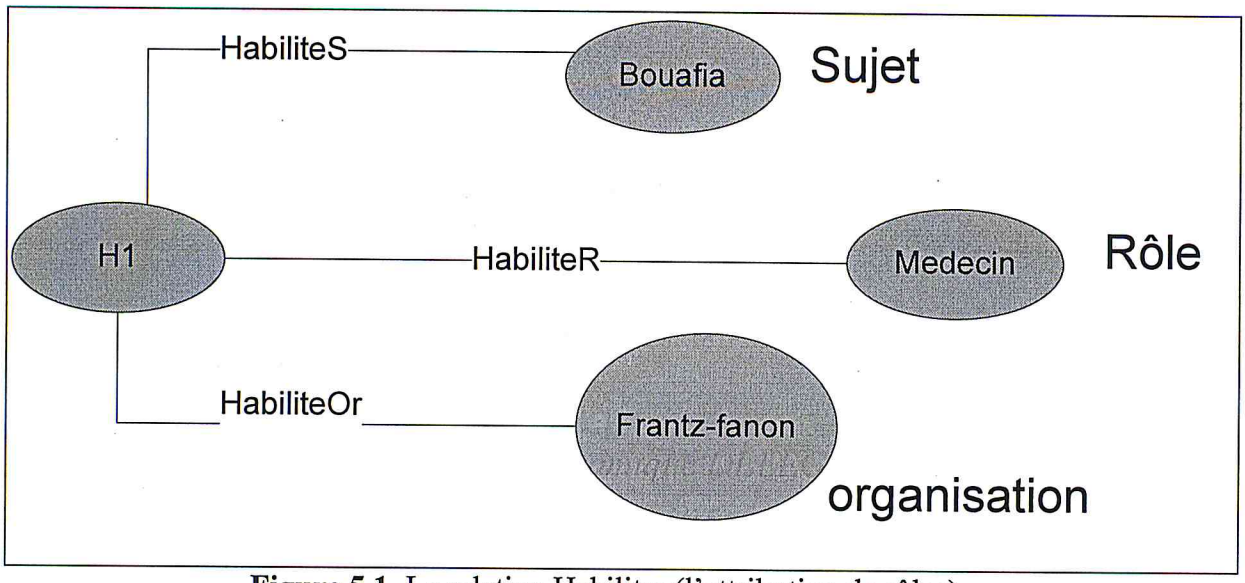

Figure 5.1. La relation Habiliter (l'attribution de rôles).

Exemple :

La connaissance "*Bouafia* " est un sujet habilité au rôle "*Medecin* " dans l'organisation "*Frantz-Fanon*" est représentée par H1 qui est une instance du concept "*Habilite*" (**Figure 5.1**).

La description est comme suit :

Habilite (H1) ⊑ HabiliteS.Sujet (Bouafia) ⊓ HabiliteR.Role (Médecin) ⊓

HabiliteOr.Organisation (Frantz-Fanon).

3.1.2. Axiome de définition de vue :

Nous avons :

Objet ⊑ T

Vue ⊑ T

Organisation ⊑ T

Utilise ⊑ UtiliseO.Objet ⊓ UtiliseV.Vue ⊓ UtiliseOr.Organisation

Cet axiome définit la relation entre un objet, une vue et une organisation en utilisant les relations binaires : UtiliseO, UtiliseV, UtiliseOr tel que :

- UtiliseO: relie le concept Utilise au concept Objet ;
- UtiliseV: relie le concept Utilise au concept Vue ;
- UtiliseOr: relie le concept Utilise au concept Organisation.

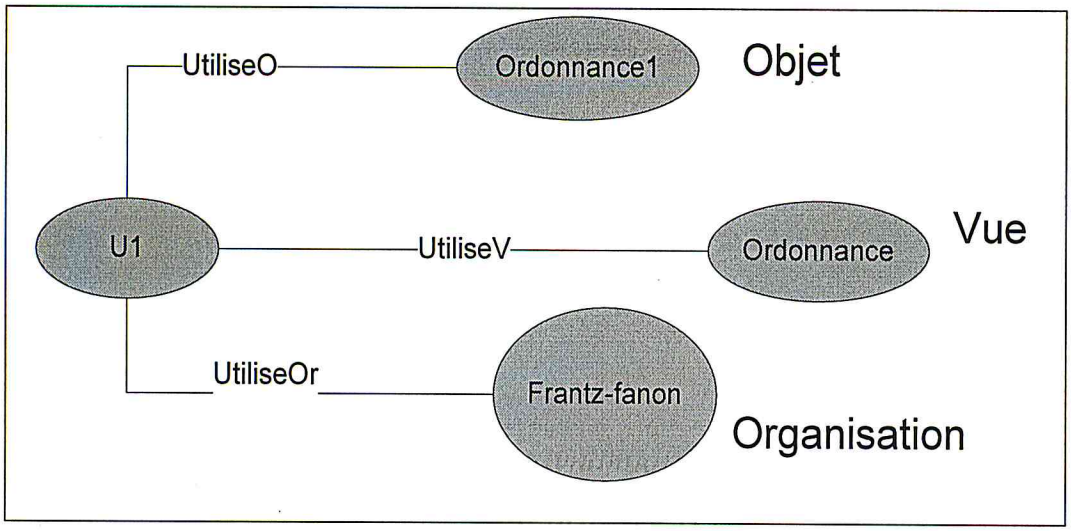

Figure 5.2. La relation Utilise (la définition de vue).

Exemple :

la connaissance "ordonnance 1" est un objet utilisé dans la vue « Ordonnance » dans l'organisation "Frantz-fanon " est représentée par U1 qui est une instance du concept "Utilise" (**Figure 5.2**).

- La description est comme suit :

Utilise (U1) ⊑ UtiliseO.Objet (Ordonnance 1) ⊓ UtiliseV.Vue (Ordonnance) ⊓ UtiliseOr.Organisation (Frantz-fanon).

3.1.3. Axiome de définition d'activités :

Nous avons :

Action ⊑ T

Activité ⊑ T

Organisation ⊑ T

Considere ⊑ ConsidereAc.Action ⊓ ConsidereAv.Activite ⊓ ConsidereOr.Organisation

Cet axiome définit la relation entre une action, une activité et une organisation en utilisant les relations binaires : ConsidereAc, ConsidereAv, ConsidereOr tel que :

- ConsidereAc: relie le concept Considère au concept Action ;
- ConsidereAv: relie le concept Considère au concept Activité ;
- ConsidereOr: relie le concept Considère au concept Organisation.

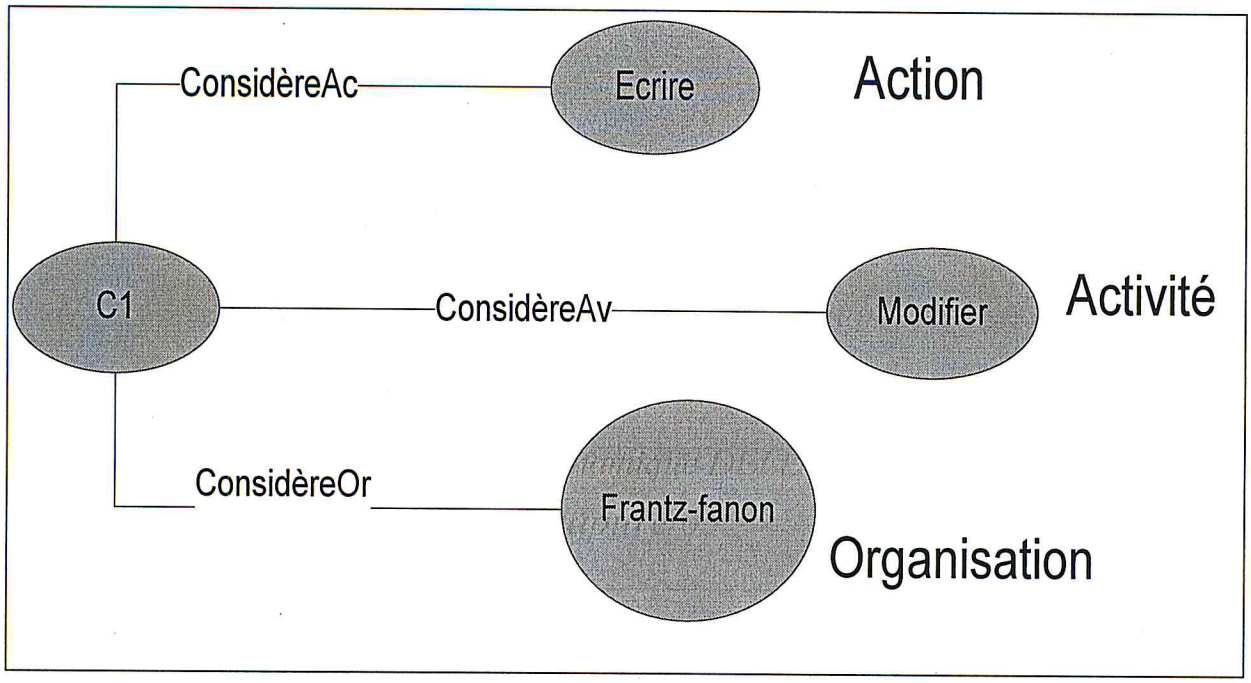

Figure 5.3. La relation Considère (Définition d'activité)

Exemple :

la connaissance "Ecrire" est une action utilisé dans l'activité "Modifier" dans l'organisation "Frantz-fanon " est représentée par C1 qui est une instance du concept "Considère" (**Figure 5.3**).

- La description est comme suit :

Considere (C1) ⊑ ConsidereAc.Action (Ecrire) ⊓ ConsidereAv.Activite (Modifier) ⊓ ConsidereOr.Organisation (Frantz-fanon).

3.1.4. Axiome de définition de hiérarchies :

Notre système supporte une architecture à trois types de hiérarchies : Hiérarchie de rôles, de vues, et d'activités. Nous décrivons les axiomes de définition de chacune de ces hiérarchies dans ce qui suit .

3.1.4.1. Axiome de définition de hiérarchie de rôle :

La description relative à la définition de la hiérarchie de rôles est la suivante :

Sous-rôle ⊑ Sous- rôle 1. Rôle ⊓ Sous- rôle 2. Rôle ⊓ Sous- rôleOr.Organisation

Le rôle 1 est un sous rôle du rôle 2 dans l'organisation, on implémente les relations binaires Sous- role1, Sous- role2 et Sous- roleOr de telle façon que :

- **Sous- rôle1** : relie le concept Sous- rôle au concept Rôle ;
- **Sous- rôle2** : relie le concept Sous- rôle au concept Rôle ;

- **Sous- rôleOr** : relie le concept Sous- rôle au concept Organisation.

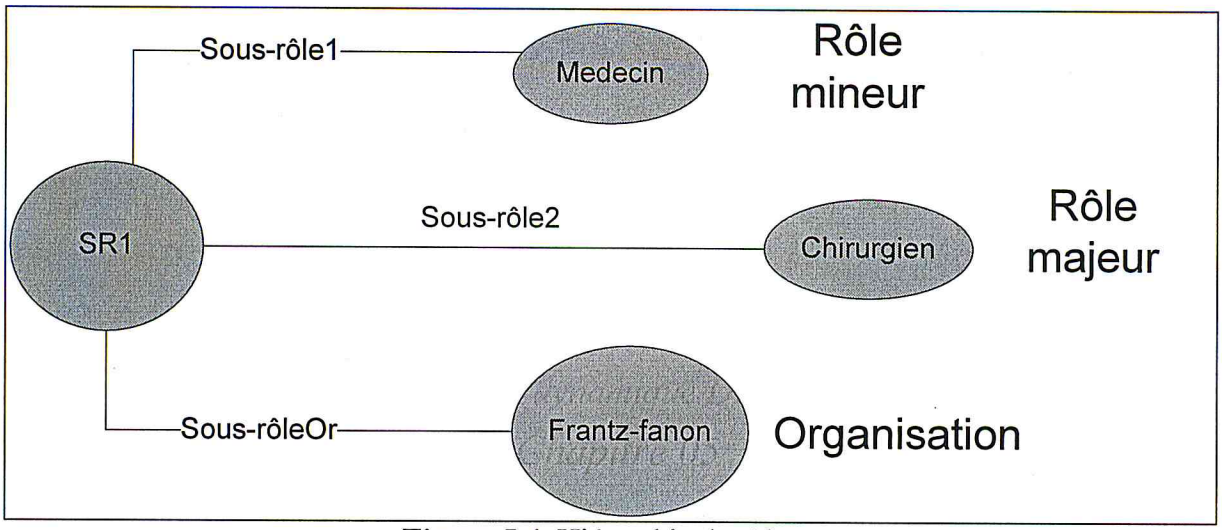

Figure 5.4. Hiérarchie de rôles

Exemple :

Pour spécifier que le rôle « Chirurgien» est un sous rôle du rôle "Médecin " dans l'organisation "Frantz-fanon ", nous définissons SR1 qui est une instance de Sous-rôle (**Figure 5.4**) .

- La description est comme suit :

Sous-rôle(SR1) ⊑ Sous-rôle1.Rôle (Chirurgien) ⊓ Sous- rôle2.Rôle (Médecin) ⊓ SousrôleOr.Organisation (Frantz-fanon).

3.1.4.2. Axiome de définition de hiérarchie d'activités :

La description relative à la définition de hiérarchie d'activités est la suivante :

Sous-activité ⊑ Sous-activité1.Activité ⊓ Sous-activité2.Activité ⊓ Sous-activitéOr.Organisation

L'activité 1 est une sous activité de l'activité 2 dans l'organisation Organisation ou, on Implémente les relations binaires Sous-activite1, Sous-activite2 et Sous-activiteOr de telle façon que :

- **Sous- activité1** : relie le concept Sous- activité au concept Activité ;
- **Sous- activité2** : relie le concept Sous- activité au concept Activité ;
- **Sous- activitéOr** : relie le concept Sous- activité au concept Organisation.

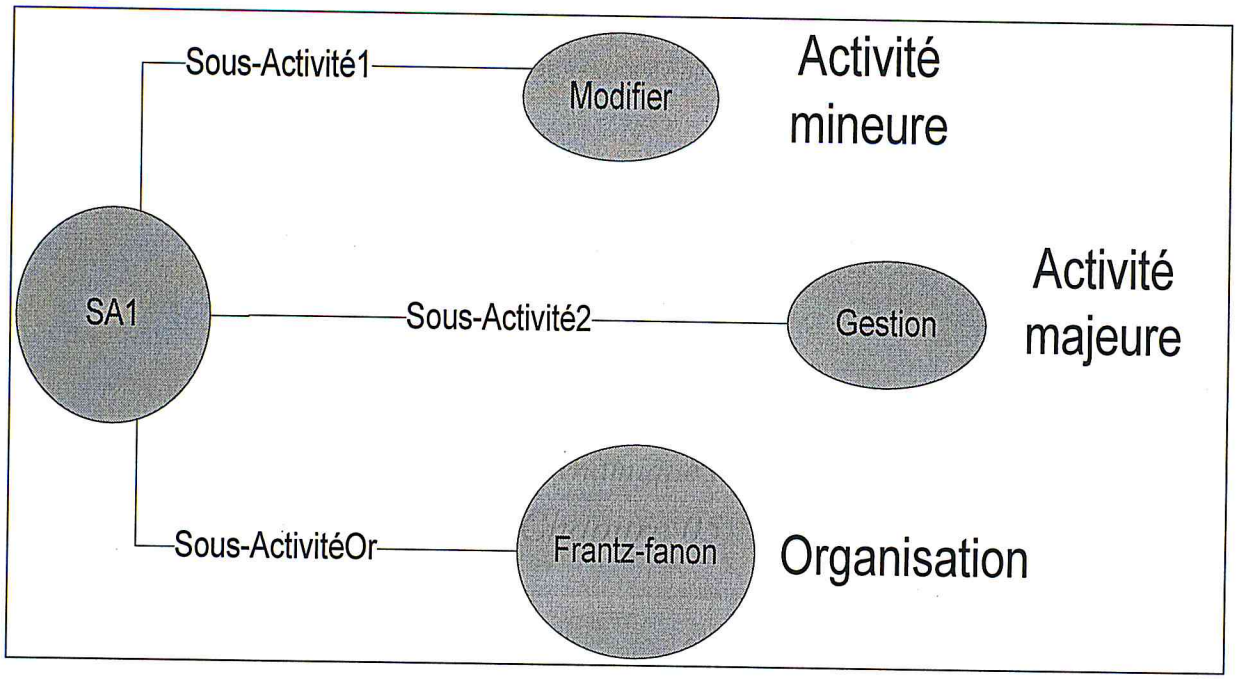

Figure 5.5. Hiérarchie d'activités.

Exemple :

Pour spécifier que l'activité "Modifier" est une sous activité de l'activité

"Gestion" dans l'organisation "Frantz-Fanon", nous définissons SA1 qui est une instance de Sousactivité (**Figure 5 .5**)

- La description est comme suit :

Sous- activité(SA1) ⊑ *Sous- activité1.Activité (Modifier)* ⊓ *Sous-activité2.Activité (Gestion)*⊓ *Sous- activitéOr.Organisation (Frantz-fanon).*

3.1.4.3. Axiome de définition de hiérarchie de vues :

La description relative à la définition de la hiérarchie de vues est la suivante :

Sous-vue ⊑ Sous- vue1.Vue ⊓ Sous- vue2.Vue ⊓ Sous- vueOr.Organisation

La vue 1 est une sous vue de la vue 2 dans l'organisation Organisation ou, on implémente

Les relations binaires Sous- vue1, Sous- vue2 et Sous- vueOr de telle facon que :

- **Sous- vue1** : relie le concept Sous- vue au concept Vue ;
- **Sous- vue2** : relie le concept Sous- activité au concept Vue ;
- **Sous- vueOr** : relie le concept Sous- vue au concept Organisation.

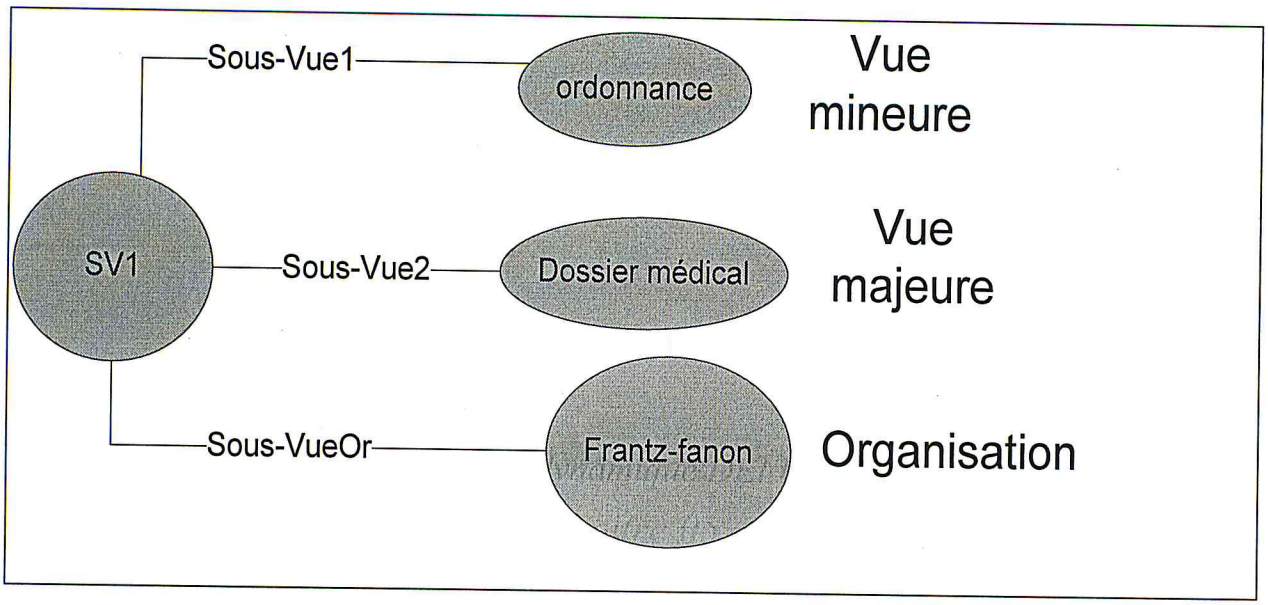

Figure 5.6. Hiérarchie de vues.

Exemple :

Pour spécifier que la vue « ordonnance » est une sous vue de la vue «Dossier médical » dans l'organisation « Frantz-Fanon », nous définissons SV1 qui est une instance de Sous- vue (**Figure 5.6**)

- La description est comme suit :

Sous-vue(SV1) ⊑ Sous-vue1.Vue(ordonnance) ⊓ Sous-vue2.Vue (Dossier médical) ⊓ SousvueOr.Organisation (Frantz-fanon).

3.1.5. Axiome d'attribution de permissions abstraites :

Cet axiome définit une relation entre les entités abstraites : organisation, rôle, activité et vue. Une permission par défaut est donnée à un rôle R dans une organisation Or pour Effectuer l'activité A sur la vue V. dans un contexte exceptionnel on génère une exception sur cette Permission qui annulera son accord.

- δ Permission ⊑ PermissionR.Role ⊓ PermissionAv.Activite ⊓ PermissionV.Vue ⊓ PermissionOr.Organisation. ➔dans le cas défaut .
- Permission ε ⊑ PermissionR.Role ⊓ PermissionAv.Activite ⊓ PermissionV.Vue ⊓ PermissionOr.Organisation➔ dans le cas exception.

Ou PermissionR, PermissionAv, PermissionV et PermissionOr sont des relations binaires tel que :

- **PermissionR :** relie le concept Permission au concept Rôle ;
- **PermissionAv** : relie le concept Permission au concept Activité ;

- **PermissionV** : relie le concept Permission au concept Vue ;
- **PermissionOr** : relie le concept Permission au concept Organisation.

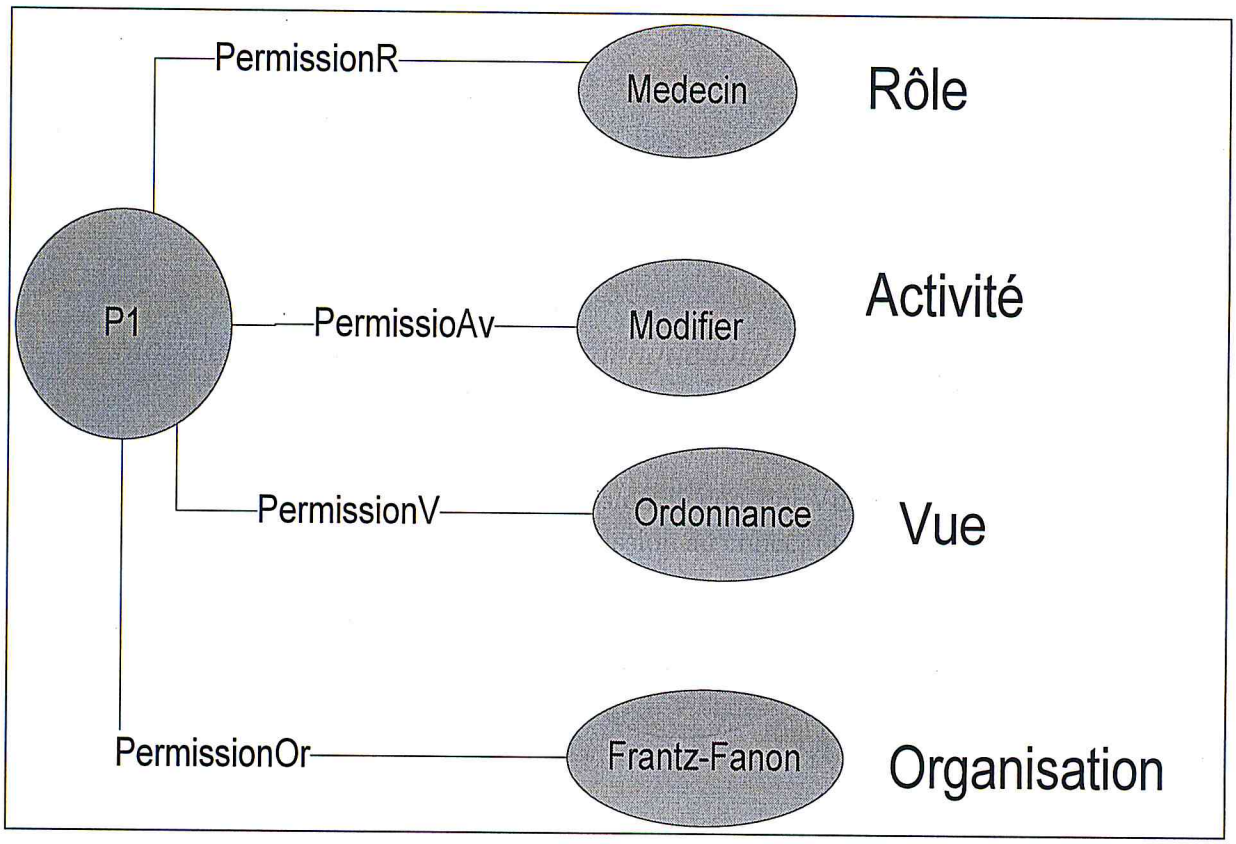

Figure 5.7. Permission abstraite.

Exemple :

Dans l'organisation "Frantz-Fanon", toute personne qui joue le rôle "Medecin" a le droit d'effectuer l'activité "Modifier " sur la vue " Ordonnance **" (Figure 5.7).** Ceci est représenté par P1 qui est une instance du concept "Permission" décrit par l'expression suivante :

δPermission(P1) ⊑ PermissionR.Rôle(Medecin) ⊓ PermissionAv.Activité(Modifier) ⊓ PermissionV.Vue(Ordonnance) ⊓ PermissionOr.Organisation(Frantz-Fanon).

3.1.6. Axiome d'attribution de permissions concrètes :

Une permission concrète implémente les entités concrètes de notre système qui sont :

Les sujets, les actions et les objets. Elle est exprimée avec l'axiome suivant :

- δEst-Permis ⊑ Est-PermisS.Sujet ⊓ Est-PermisAc.Action ⊓ Est-PermisO.Objet. ➔ dans le cas défaut .

- Est-Permis ε ⊑ Est-PermisS.Sujet ⊓ Est-PermisAc.Action ⊓ Est-PermisO.Objet. ➔ dans le cas exception.

Ou, Est-PermisS, Est-PermisAc et Est-Permis sont des relations binaires tel que :

- **Est-PermisS** : relie le concept Est-Permis avec le concept Sujet ;
- **Est-PermisAc** : relie le concept Est-Permis avec le concept Action ;
- **Est-PermisO** : relie le concept Est-Permis avec le concept Objet.

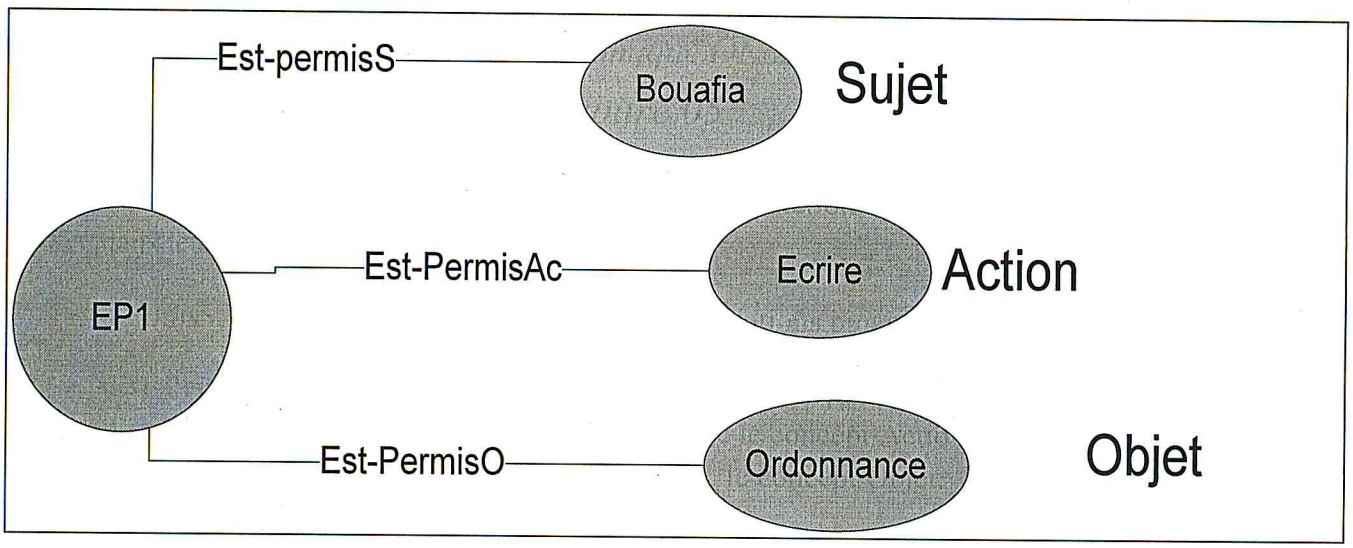

Figure 5.8. Permission concrète.

Exemple :

(La Figure 5.8) illustre le cas d'une permission concrète dans laquelle nous spécifions que le sujet « Bouafia » a la permission de faire l'action « Ecrire » sur l'objet « ordonnance ».

Nous définissons EP1 une instance du concept Est-Permis et nous écrivons :

δEst-Permis(EP1) ⊑ Est-PermisS.Sujet (Bouafia) ⊓ Est-PermisAc.Action (Ecrire) ⊓ Est-PermisO.Objet (ordonnance).

3.1.7. Axiome d'attribution de délégations :

3.1.7.1. Les Délégations dans AdOrbac :

L'approche que nous suggérons pour gérer les délégations est d'utiliser les notions de vues administratives qui sont définies dans AdOrbac**(chapitre 2).**Les vues de délégation hériteront ainsi des attributs de ces vues la comme il est montré dans **(la Figure 5.9).**

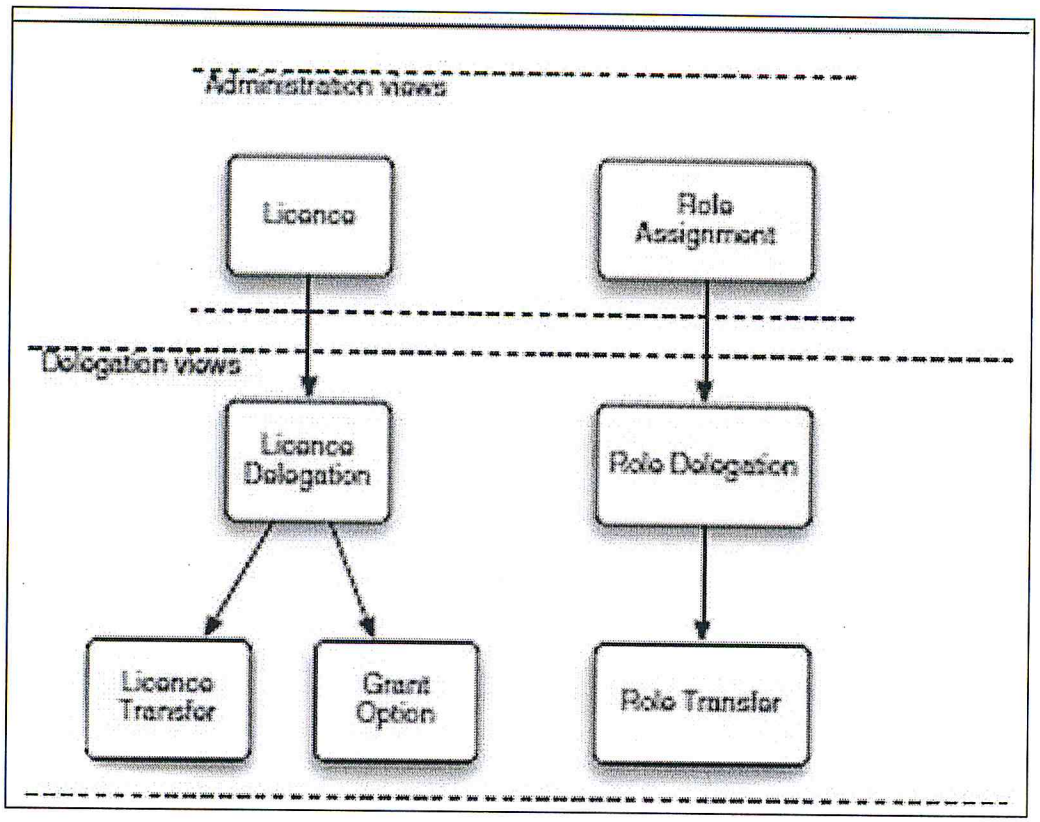

Figure 5.9 :l'héritage entre les vues administratives et les vues de délégation. [7]

La vue licence-délégation et la vue rôle-délégation sont les principales vues de la délégation. Ces vues sont utilisées respectivement pour déléguer des droits (délégation partielle et délégation totale), elles sont définies dans ce qui suit :

3.1.7.1.1. La vue licence délégation :

- **Axiome licence délégation dans le cas défaut :**

  δ Permission ⊑ UtiliseL.Licence_Délégation ⊓ BénéficiaireL.bénéficiaire ⊓ PrivilègeL.Action⊓ CibleL.Objet .

  Informellement, si (L) appartient à la vue licence délégation :
  - (bénéficiaire) est le bénéficiaire de (L),
  - (Action) est l'action permise par (L),
  - (Objet) est l'objet auquel la licence accorde l'accès,

  Alors le bénéficiaire(bénéficiaire) a la permission d'effectuer l'action (Action) sur l'objet (Objet) dans un contexte défaut .

- **Axiome licence délégation dans le cas exception :**

Permission ε ⊑ UtiliseL.Licence_Délégation ⊓ BénéficiaireL.bénéficiaire ⊓ PrivilègeL.Action⊓ CibleL.Objet .

Informellement, si (L) appartient à la vue licence délégation :

- (bénéficiaire) est le bénéficiaire de (L),
- (Action) est l'action permise par (L),
- (Objet) est l'objet auquel la licence accorde l'accès,

Alors le bénéficiaire(bénéficiaire) ne possède plus la permission d'effectuer l'action (Action) sur l'objet (Objet) puisque l'exception prédomine sur le défaut donc la permission définit dans le cas défaut s'annule si on lui génère une exception .

3.1.7.1.2. La vue rôle délégation :

- **Axiome rôle délégation :**

Habilite ⊑ *UtiliseRD.Rôle_Délégation* ⊓ *AssigneeRD.bénéficiaire* ⊓ *AssignmentRD.Rôle.*

Informellement, si (RD) appartient à la vue rôle_délégation :

- (RD) est assigné au bénéficiaire (bénéficiaire)
- (Rôle) est le rôle qui est assigné par (RD)

alors le bénéficiaire (bénéficiaire) est habilité à faire le Rôle (rôle).

Remarque

Les objets qui appartiennent aux deux vues *licence_délégation* et *rôle_délégation* ont les mêmes attributs, que les objets qui appartiennent aux deux vues *licence et rôle_assignment(voir **chapitre 2**) et* (**Figure 5.9**), sauf qu'ils ont un attribut additionnel appelé cessionnaire*.

L'insertion d'un objet dans la vue *licence_délégation* et dans la vue *rôle_délégation* permettra au cessionnaire de déléguer des permissions à un bénéficiaire Donc pour gérer la Politique de délégation nous sommes obligés de nommer le cessionnaire (délégant) qui aura un accès à ces vues dans un certain contexte.

**Cessionnaire :* est un sujet ou un rôle qui délègue une licence, c'est le délégant.

La permission du cessionnaire est définit par l'axiome suivant :

- **Axiome de définition Permission cessionnaire dans un contexte défaut**

δ permission ⊑ cessionnaireL.cessionnaire ⊓ utiliseL.licence_delegation

Informellement, le cessionnaire (cessionnaire) a la permission de déléguer une licence dans le contexte défaut .

- **Axiome de définition Permission cessionnaire dans un contexte exception**

Permission ε ⊑ cessionnaireL.cessionnaire ⊓ utiliseL.licence_delegation

Informellement, le cessionnaire (cessionnaire) perd la permission de déléguer une licence dans le contexte exception si on génère l'exception sur la permission défaut précédente .

3.1.7.2. La notion de sous licence :

- **Définition de la sous licence dans le cas défaut :**

δ Sous_Licence ⊑ CibleL.Objet ⊓ CibleL'.Objet' ⊓ Sous_CibleT.Objet' ⊓ PrivilègeL.Action ⊓ PrivilègeL'.Action' ⊓ Sous_PrivilègeL.Action' Sous_Cible ⊑ Sous_VuesO.Objet' ⊓ UtiliseO.O'.

Informellement, si (objet) est l'objet auquel la licence (L) a accordé l'accès,

- (objet') est l'objet auquel la licence (L') a accordé l'accès,
- (objet) est le sous objet de (objet') ,
- (action) est l'action permise par (L) ,
- (action') est l'action permise par (L'),
- (P) est la sous action de (P'),

Alors (L) est une sous licence de (L') dans dans un contexte défaut.

- **Définition de la sous licence dans le cas exception :**

Sous_Licence ε ⊑ CibleL.Objet ⊓ CibleL'.Objet' ⊓ Sous_CibleT.Objet' ⊓ PrivilègeL.Action ⊓ PrivilègeL'.Action' ⊓ Sous_PrivilègeL.Action' Sous_Cible ⊑ Sous_VuesO.Objet' ⊓ UtiliseO.O'.

Informellement, si (T) est l'objet auquel la licence (L) a accordé l'accès,

- (objet') est l'objet auquel la licence (L') a accordé l'accès,
- (objet) est le sous objet de (objet') ,
- (action) est l'action permise par (L) ,
- (action') est l'action permise par (L'),
- (P) est la sous action de (P'),

Alors (L) n'est plus une sous licence de (L') dans un contexte exception, si cette exception est générée sur le défaut précédent.

Remarque :

- Nous considérons (O) est le sous objet de (O') si seulement si nous avons deux vues (V) et (V') tel que (V) est la sous vue de (V') ou si l'objet (O) est utilisé dans la vue (V').

Sachant que :

O'⊑ Objet.

- le prédicat équivalence_licence est défini comme suit :

Informellement, si (L) est une sous licence de (L') et (L') est une sous licence de (L) alors la licence (L) est équivalente à la licence (L').

Équivalence_Licence ⊑ Sous_LicenceL.L'⊓ Sous_LicenceL'.L.

3.1.7.3. Définition des types de la Délégation :

Dans cette partie nous allons définir une partie des différents types de délégation(les plus importants), nommés dans un chapitre précédent **(chapitre 2)** :

3.1.7.3.1 Délégation Permanente/temporaire :

La délégation temporaire étant une exception de la délégation permanente, nous avons choisi de modéliser ces notions en disant qu'une délégation est permanente jusqu'à nouvel ordre quand on veut l'interrompre on génère une exception dessus ce qu'il la rendra ainsi temporaire .Donc ce type de délégation sera modélisé implicitement.

3.1.7.3.2 Délégation Monotone/non monotone :

Pour modéliser la délégation non-monotone, une licence_transfer est définie comme suit :

- **Définition de la vue licence_transfer dans le contexte défaut:**

δ Licence_Délégation ⊑ UtiliseL.licence_transfer.

Informellement, si (L) appartient à la vue licence_délégation alors (L) appartient aussi à la vue licence_transfer dans un contexte défaut (où le type de délégation est non monotone).

La délégation monotone est considérée comme une exception de la délégation non monotone donc on aura :

- **Définition de la vue licence_transfer dans le contexte exception :**

Licence_Délégation ε⊑ UtiliseL.licence_transfer

Informellement, si (L) appartient à la vue licence_délégation alors (L) ne vas plus appartenir aussi à la vue licence_transfer dans un contexte exception (où le type de délégation est

monotone)puisque dans une délégation monotone il n'y plus de transfert mais on conserve le droit qu'on délègue .

3.1.7.3.3. Délégation à n-pas :

Cette caractéristique définit si le cessionnaire peut ou pas déléguer une permission et ça par rapport au niveau hiérarchique de son rôle dans l'organisation. La vue grant_option_licence est définie comme suit :

- **Axiome de Permission dans le cas de la vue grant_option_licence dans un contexte défaut :**

δ *Permission* ⊑ *UtiliseL.GOL* ⊓ *BénéficiaireL.bénéficiaire* ⊓ PrivilègeL.Action ⊓ *CibleL.Objet* ⊓ *niveau* <=*niveauMax*

Informellement, si (L) appartient à la vue grant_option_licence(GOL),

- (bénéficiaire) est le bénéficiaire de (L),
- (Objet) est l'objet auquel (L) permet l'accès,
- Niveau qui reste inférieur au niveau défini par l'administrateur (niveau max) et qui se décrémenté à chaque fois qu'il y 'a eu une délégation .
  alors le bénéficiaire (bénéficiaire) a la permission de déléguer une Licence dans un dans un contexte défaut si son niveau reste inférieur au niveau max.
  le niveau max que nous avons défini est de 3 .

- **Axiome de Permission dans le cas de la vue grant_option_licence dans un contexte exception :**

Permissionε ⊑ *UtiliseL.GOL* ⊓ *BénéficiaireL.bénéficiaire* ⊓ PrivilègeL.Action ⊓ *Cible.Objet* ⊓ *niveau* <=*niveauMax*

Informellement, si (L) appartient à la vue grant_option_licence(GOL),

- (bénéficiaire) est le bénéficiaire de (L),
- (Objet) est l'objet auquel (L) permet l'accès,
- (Action) est l'action permise par (L) d'être effectuée
- Niveau qui reste inférieur au niveau défini par l'administrateur (niveau max) et qui se décrémente à chaque fois qu'il y 'a eu une délégation .
  alors le bénéficiaire (bénéficiaire) perds la permission de déléguer une Licence dans le contexte exception.

3.1.7.3.4 .Révocation :

La révocation est un aspect important dans les modèles de délégation. Dans cette section, nous présenterons certaines propriétés de la révocation:

- ❖ Dans le cas de, Grande_Dépendance_Révocation (GD) le cessionnaire a le droit de révoquer la licence d'une délégation ou d'un rôle.
- ❖ Dans le cas de, Grande_ Indépendance _Révocation (GID) n'importe quel membre qui a ce droit a le droit de révoquer le droit au bénéficiaire.

Ceci est modélisé dans ce qui suit :

- ***Le cas GD est défini comme suit :***

Définit ⊑ UtiliseL.Licence_Délégation ⊓ CessionnaireL.cessionnaire.

Informellement, si (L) appartient à la vue licence_ délégation et (cessionnaire) est le cessionnaire de (L) *alors le cessionnaire détient l'autorisation de révoquer (L)* .

- ***Le cas GID est défini comme suit :***

Définit ⊑ UtiliseL.Licence_Délégation ⊓ CessionnaireL.cessionnaire ⊓ HabiliteGR.Rôle⊓ HabiliteU.Rôle.

Informellement, si (L) appartient à la vue licence_ délégation,

- (cessionnaire) est le cessionnaire de (L),
- (GR) est habilité de faire le Rôle,
- (U) est habilité de faire le Rôle alors l'utilisateur (U) détient l'autorisation de révoquer (L).

3.1.7.2.4.1 Révocation en cascade :

La délégation du droit n'est valide que si le délégant possède encore ce droit.

Notons que la vue *délégation_en_cascade* est une sous vue de la vue *licence_délégation*, A cet effet, la vue *délégation_en_cascade* est définie comme suit :

- **Axiome définition de vue délégation_en_cascade dans le cas défaut :**

δ Permission ⊑ UtiliseL.Délégation_En_Cascade ⊓ BénéficiaireL.benéficiaire ⊓ CessionnaireL.cessionnaire ⊓ PrivilègeL.Action ⊓ CibleL.Objet

Informellement, si (L) est une licence qui appartient à la vue délégation_en_cascade(DEC) :

- (bénéficiaire) est le bénéficiaire de (L),
- (cessionnaire) est le délégant de (L),
- (Objet) est l'objet auquel (L) permet l'accès,
- (Action) est l'action permise par (L) d'être effectuée

Alors le bénéficiaire (cessionnaire) qui aussi le déléguant de (L) a la permission de faire l'action (Action) sur l'objet (Objet) dans le contexte défaut .

- **Axiome définition de vue délégation_en_cascade dans le cas exception :**

Permission ε ⊑ UtiliseL.Délégation_En_Cascade ⊓ BénéficiaireL.bénéficiaire⊓ CessionnaireL.cessionnaire ⊓ PrivilègeL.Action ⊓ CibleL.Objet

Informellement, si (L) est une licence qui appartient à la vue délégation_en_cascade(DEC) :

- (bénéficiaire) est le bénéficiaire de (L),
- (cessionnaire) est le délégant de (L),
- (Objet) est l'objet auquel (L) permet l'accès,
- (Action) est l'action permise par (L) d'être effectuée
  Alors le bénéficiaire (cessionnaire) qui aussi le déléguant (L) perds la permission de faire l'action (Action) sur l'objet (Objet) l dans le contexte exception.

➢ Donc la révocation en cascade consistera à supprimer les objets qui appartiennent à la vue délégation en cascade.

3.1.8. Définition des règles de sécurité :

Afin de compléter la spécification de la politique, il est important d'exprimer un ensemble de règles visant à définir comment le système peut évoluer sans compromettre les objectifs de sécurité identifiées.

Nous définissons cas de règles de sécurités dans notre modèle.

3.1.8.1. Règle de sécurité dans le cas de défaut :

δ Est-Permis ⊑ δ Permission ⊓ Habilite ⊓ Utilise ⊓ Considère

Si :

- ✓ Une permission par défaut existe pour un rôle R, une activité Av et une vue V dans une organisation Or (δ Permission) .
- ✓ Un sujet S est habilité dans ce rôle R (Habilite) .
- ✓ Un objet O est utilisé dans cette vue V (Utilise) .
- ✓ Une action Ac implémente cette activité Av (Considère).

Alors Le sujet S est par défaut permis d'effectuer l'action Ac sur l'objet O (δ Est- Permis).

Etant donné que Est-Permis ⊑ δ Est-Permis (une permission concrète peut être déduite D'une permission par défaut), on peut dire finalement que le sujet S a la permission D'effectuer l'action Ac sur l'objet O.

3.1.8.2. Règle de sécurité dans le cas d'exception :

Est-Permis ε ⊑ Permission ε ⊓ Habilite ⊓ Utilise ⊓ Considère

Dans le cas où on a une exception sur un concept permission (Permission ε), on dit qu'on a une exception sur le concept Est-Permis noté Est-Permis ε. On sait que Est- Permis ε (une permission concrète ne peut être déduite d'une permission exceptionnelle), de cela on déduit que le sujet S n'est pas permis d'effectuer l'action Ac sur l'objet O.

3.2.ABox :

Dans notre cas, la ABox va contenir les entités concrètes qui sont l'ensemble des sujets de l'organisation, par exemple, Amina , Ali, Mohamed , etc., l'ensemble des actions des activités de l'organisation, prenons comme exemple, lire, écrire, supprimer, etc. et tous les objets des vues introduites dans l'organisation, par exemple, dossier médical du patient, ordonnance, etc. Nous donnerons plus de détails dans le chapitre suivant.

3.3. Mécanismes d'inférences :

Dans notre modèle le principal mécanisme d'inférence utilisé est l'héritage (héritage de permissions dans la hiérarchie de rôles, d'activités et de vues).

Nous donnerons plus de détails dans le chapitre suivant.

4. Conclusion :

Dans ce chapitre, nous avons montré la modélisation du modèle de contrôle d'accès dynamique et contextuel DELEGATION-OrBAC δε à travers la définition de la TBOX , de ce que va contenir la ABOX et des mécanismes d'inférences traités , la réalisation de ce modèle sera présenté dans le chapitre suivant .

Chapitre 06

Réalisation du Modèle DELEGATION_OrBACδε

Chapitre 06

1. Introduction :

Nous décrivons dans ce chapitre l'environnement dans lequel nous avons réalisé le modèle DELEGATION-OrBACδε, l'implémentation que nous avons proposée, qui est un système d'information médical, ainsi que la description des différentes étapes qui montrent le fonctionnement de ce système.

2. Outils de développement :

Notre travail a été conçu dans un environnement Windows 8, en utilisant le langage de programmation java sous Netbeans IDE 7.3 pour l'implémentation des différents mécanismes traités dans les chapitres précédents ainsi que pour la réalisation de l'interface graphique et Microsoft Office Access 2007 pour la persistance de la base de connaissances.

2.1. Brève Présentation des Outils de développement :

- **Windows 8 :**

Windows 8 est la dernière version du système 'exploitation Windows commercialisée depuis le 26 octobre 2012. La version alpha (*Developer Preview*) publique a vu le jour en septembre 2011[3], une version bêta (*Consumer Preview*) a aussi vu le jour le 29 février 2012 à l'occasion du Mobile World Congress à Barcelone. La version Release Preview (équivalent des Release Candidate, RC, des versions antérieures à Windows 8) est sortie le 31 mai 2012. Pour Microsoft, Windows 8 est devenu une priorité en juillet 2010.

Windows 8 a été dévoilé, avec l'utilisation de l'interface tactile, le 1er juin 2011. La version RTM de Windows 8 à destination des constructeurs OEM est disponible depuis le 15 août 2012. le 25 octobre 2012, était la sortie grand public le lendemain pour les différentes plates-formes (tablettes, PC et smartphones).**[22]**

- **Netbeans IDE 7.3 :**

NetBeans IDE est un environnement de développement qui s'adapte aux langages de programmation (Javascript, Python, PHP, Groovy, C/C++, etc.), aux outils et aux ressources dont vous disposez. Le programme détecte automatiquement la présence de Java, JDK, SOA, Ruby, MySQL, etc. sur votre système, ainsi que les serveurs Apache ou GlassFish, pour vous fournir les plugins nécessaires. Vous pourrez ainsi créer facilement des applications Web, des portails d'entreprise, des logiciels multiplateformes sous Java, des logiciels pour mobiles, etc. Vous disposerez de nombreux outils d'édition, d'un debugger, d'une module de prévisualisation, de modèles et de bibliothèques, de fonctions de complétion et d'implémentation, etc. La dernière version de NetBeans IDE supporte dorénavant les spécifications du langage de développement Java SE. Il s'intègre également avec les serveurs Oracle WebLogic et supporte Oracle Database et GlassFish. On note au passage la facilité d'édition HTML5 et un nouveau GridBagLayout conçu pour améliorer le développement d'interfaces utilisateur et l'édition de code Java.**[21]**

- **Microsoft Office Access 2007** :

De nombreuses nouveautés ont été apportées par rapport aux versions précédentes, notamment à l'interface qui a été entièrement repensée afin que les utilisateurs puissent tirer facilement le maximum des différentes applications constituant la suite (Word, Excel, PowerPoint, Outlook, Access, Publisher, OneNote, Groove, InfoPath, SharePoint Designer, Visio et Project selon les déclinaisons du produit). Des assistants vous guident tout au long de la création de vos documents afin d'obtenir les meilleurs résultats rapidement. Enfin, des outils d'apprentissage en ligne sont désormais mis à disposition de la communauté afin que l'ensemble des utilisateurs puissent affiner leurs connaissances des logiciels intégrés à la suite**. [21]**

3. Exemple d'illustration :

Le système d'information médical qui est le système d'information du Service de Cardiologie et de Médecine interne de l'hôpital Frantz-Fanon de Blida est choisi comme exemple d'illustration. Nous avons procédé à l'interview du Chef de ce service afin de dresser 3 organigrammes : l'un pour les rôles présents dans le service (**Figure 6.1**), l'un pour la hiérarchie de vues (**Figure 6.2**), et l'un pour la hiérarchie d'activités (**Figure 6.3**) dans cette organisation.

Suivant les organigrammes nous avons aussi dresser les rôles et sous rôles (la hiérarchie de rôles), les vues et sous vues (la hiérarchie de vues) ainsi que les activités et sous activités (la hiérarchies d'activités)présents dans notre organisation Service de Cardiologie dans un tableau (**Tableau 6.1**).

Rôle , Activité ou Vue	**Rôle , Activité ou Vue Supérieur(e)**
Rôle (Chef de Service)	Ne possède pas de Supérieur.
Rôle (Infirmier)	Personnel Paramédical
Rôle (Secrétaire)	Personnel technique et administratif
Rôle (Secrétaire chargée d'hospitalisation)	Personnel technique et administratif
Rôle (Maitre-Assistant)	Personnel Médical
Rôle (Assistant)	Personnel Médical
Rôle (Chef d'unité) ,	Personnel Médical
Rôle (Résidant)	Personnel Médical
Rôle (Assistant)	Personnel Médical
Rôle (Aide-Soignante)	Personnel Paramédical
Vue (Dossier médical)	Ne Possède pas de Supérieure
Vue (fiche information)	Dossier administratif
Vue (Dossier Administratif)	Ne possède pas de Supérieure
Vue (Bilan)	Dossier Médical
Vue(ECOGRAPHIE-DOPPLER)	Dossier Médical
Vue(Electro-cardiogramme)	Dossier Médical
Vue (Observation)	Dossier Médical
Vue (imagerie),	Dossier Médical
Vue (Diagnostic) ;	Dossier Médical
Vue (Histoire du Malade)	Dossier Médical
Vue (Demande d'hospitalisation)	Dossier Administratif
Vue (Dossier médical)	Dossier médical
Activité (activité générale)	Ne possède pas de Supérieure
Activité (Prescrire)	Activité générale
Activité (Consulter)	Activité générale
Activité (Hospitaliser)	Activité générale
Activité (Gérer) ;	Activité générale
Activité (Gestion administrative) ;	Gérer
Activité (Gestion médicale) ;	Gérer
Activité (Analyser) ;	Consulter
Activité (Diagnostiquer)	Consulter

Tableau 6.1 : tableau résumant la hiérarchie dans l'organisation.

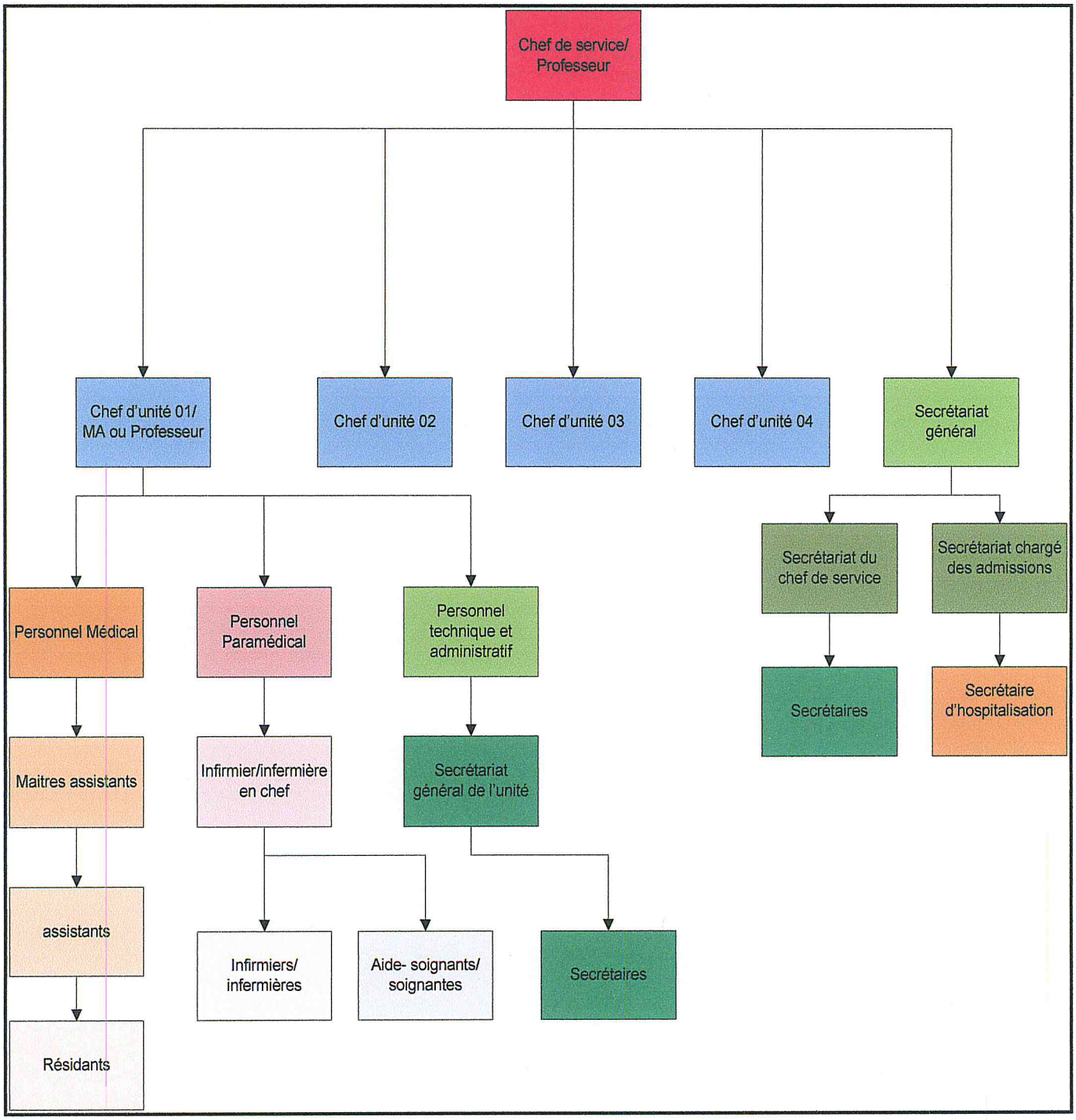

Figure 6.1 : rôles présents dans le service.

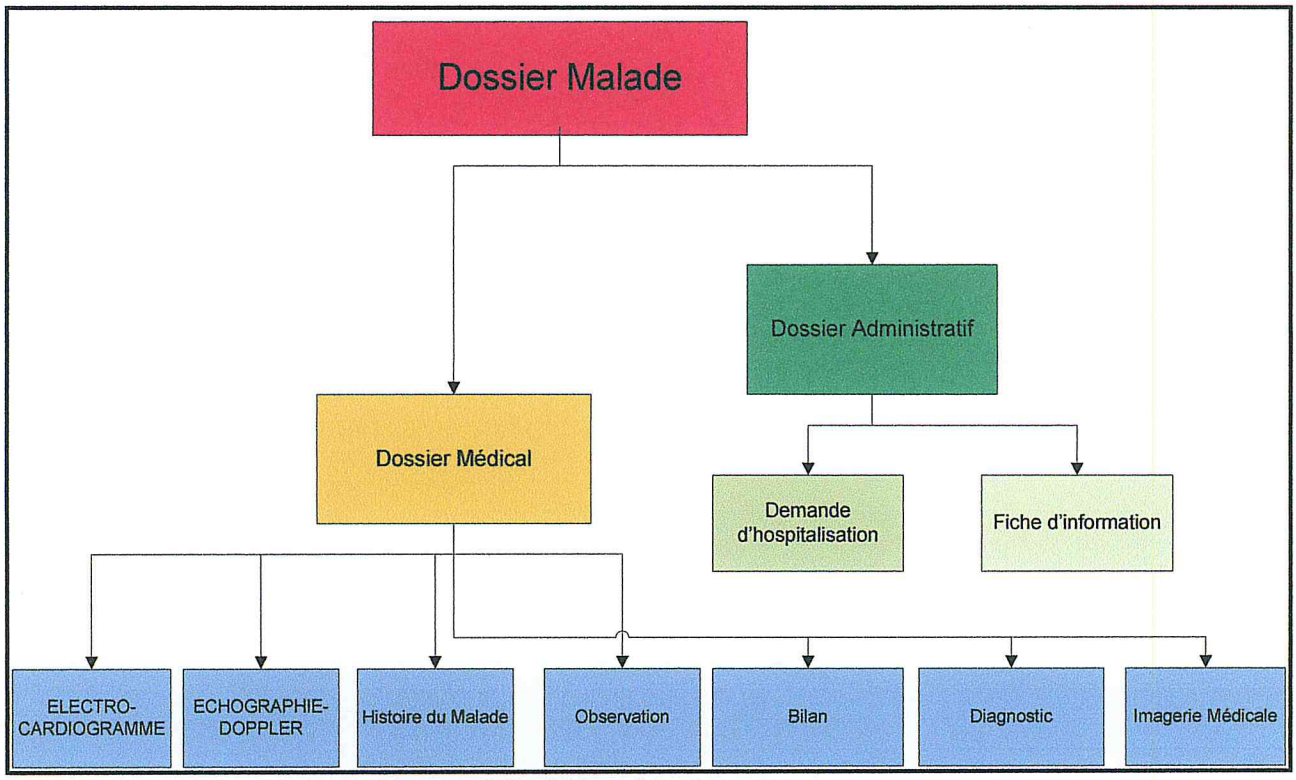

Figure 6.2 : hiérarchie de vues.

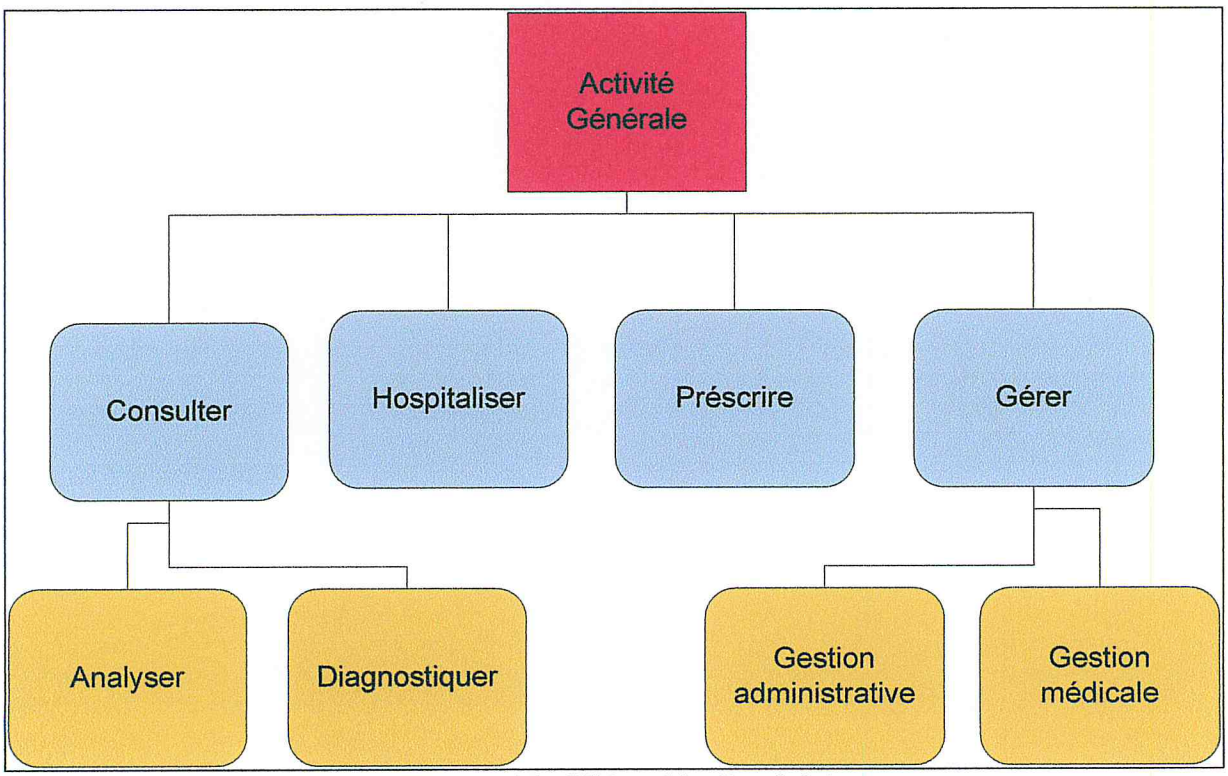

Figure 6.3 : Hiérarchie d'activités.

Remarque :

Pour le nom du sujet nous prenons par défaut le nom de famille de l'individu en question.

- De ce qui précède on peut dresser dans ce qui suit une liste de toutes les entités composants notre A-BOX et grâce à l'interview du chef de service de cardiologie de l'hôpital Frantz-Fanon nous avons pu dresser les instances des entités ainsi que des relations entre elles respectivement la liste des habilitations, considérations et utilisations.

Organisations
Organisation (Service de Cardiologie) ;
Rôles
Rôle (Maitre-Assistant) ; Rôle (Infirmier) ; Rôle (Secrétaire) ; Rôle (Secrétaire chargée d'hospitalisation) ;Rôle (Assistant) ; Rôle (Chef de Service) ; Rôle (Chef d'unité) , Rôle (Résidant) , Rôle (Aide-Soignante) ;
Sujets
Sujet (Bouafia) ; Sujet (Boureghda) ; Sujet (Sali) ; Sujet (Chettibi) ; Sujet (Bachir-cherif) ;Sujet(zitouni) ;Sujet(Benghezel)
Activités
Activité (activité générale) ; Activité (Prescrire) ; Activité (Consulter) ; Activité (Hospitaliser) ; Activité (Gérer) ; Activité (Gestion administrative) ; Activité (Gestion médicale) ; Activité (Analyser) ; Activité (Diagnostiquer)

Actions
Action (Lire) ; Action (Créer) ; Action (Supprimer) ; Action (Modifier) ;
Vues
Vue (Dossier médical) ; Vue (Dossier Administratif) ; Vue (fiche information) ;Vue(ECOGRAPHIE-DOPPLER) ; Vue(Electro-cardiogramme) ; Vue (Observation) ; Vue (Bilan) ; Vue (imagerie), Vue (Diagnostic) ; Vue (Dossier médical) ; Vue (Histoire du Malade) ; Vue (Demande d'hospitalisation).
Objets
Objet (Diagnostic1) ; Objet (Ordonnance1)
;Objet(HistoireduMalade1),Objet(imagemédicale1) ;
Objet(DossierM),Objet(DossierA),Objet(fiche information).

Habilite
Habilite (H1)$\sqsubseteq$ HabiliteS.Sujet(Bouafia) $\sqcap$ HabiliteR.Role (Chef de Service) $\sqcap$HabiliteOr.Organisation(Service de Cardiologie) ;
Habilite (H2) $\sqsubseteq$ HabiliteS.Sujet(Boureghda)$\sqcap$HabiliteR.Role (Chef d'unité) $\sqcap$ HabiliteOr.Organisation(Service de Cardiologie) ;

Considère
Considere (C1)⊑ ConsidereAc.Action (Lire) ⊓ ConsidereAv.Activite (Activité générale) ConsidereOr.Organisation (Service de cardiologie) ;
Considere (C2)⊑ ConsidereAc.Action (Créer) ⊓ ConsidereAv.Activite (Gérer) ConsidereOr.Organisation (Service de cardiologie) ;

Utilise
Utilise (U1) ⊑ UtiliseO.Objet (DossierM) ⊓ UtiliseV.Vue (Dossier Médical) ⊓ UtiliseOr.Organisation (Service de Cardiologie) ;
Utilise (U2) ⊑ UtiliseO.Objet (DossierA) ⊓ UtiliseV.Vue (Dossier Administratif) ⊓ UtiliseOr.Organisation (Service de Cardiologie) ;
Utilise (U3) ⊑ UtiliseO.Objet (Fiche information) ⊓ UtiliseV.Vue (Dossier Malade) ⊓ UtiliseOr.Organisation (Service de Cardiologie) ;

4. Fonctionnalités du système simulant notre modèle :

Dans cette partie nous allons nous intéresser aux fonctionnalités que **DELEGATION-OrBac *δε*** offre.

4.1. Interface principale :

Nous avons choisi d'organiser l'interface principale comme une vue d'ensemble sur notre système avec les entités dressées en catégories :sur une partie en (rôles , vues , organisation , activités , sujets , objets ,actions) et sur l'autre en (concept concret et abstrait) .aussi à la sélection d'une organisation nous pouvons visualiser les individus abstraits qui la composent, et en sélectionnant l'un deux nous pouvons visualiser les individus concrets qui les composent .Cette interface comprends aussi une barre des taches donnant accès aux autres interfaces de l'application . (**Figure 6.4**)

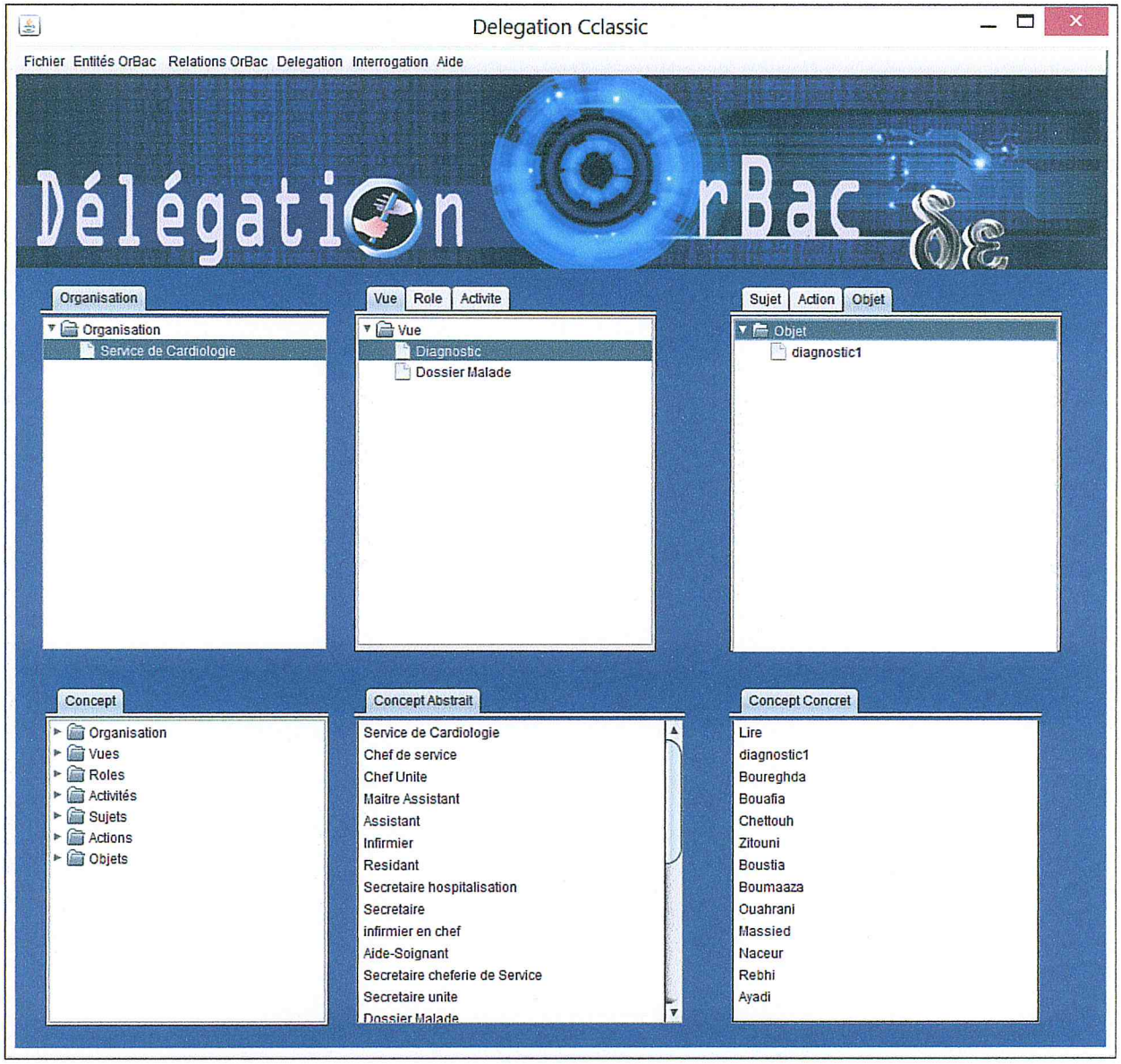

Figure 6.4 : interface principale de l'application.

4.2. Création d'une instance de concept :

Nous allons montrer comment se fait la création d'une entité à travers une section suivante (**4.5**) où nous allons présenter un exemple qui illustrera les fonctionnalités de l'application .Dans cette section nous montrons à titre d'exemple l'ajout d'un rôle suivant son supérieur puisque ceci n'est pas montré dans l'exemple qu'on proposera.

- Pour ajouter un rôle il faut aller dans le menu de l'interface principale (entités OrBAC) et cliquer sur ajouter entité du menu déroulant et ainsi sélectionner ajouter rôle.
  A l'insertion d'un rôle une boite de message apparait pour confirmer que le rôle a bien été inséré suivant son supérieur.

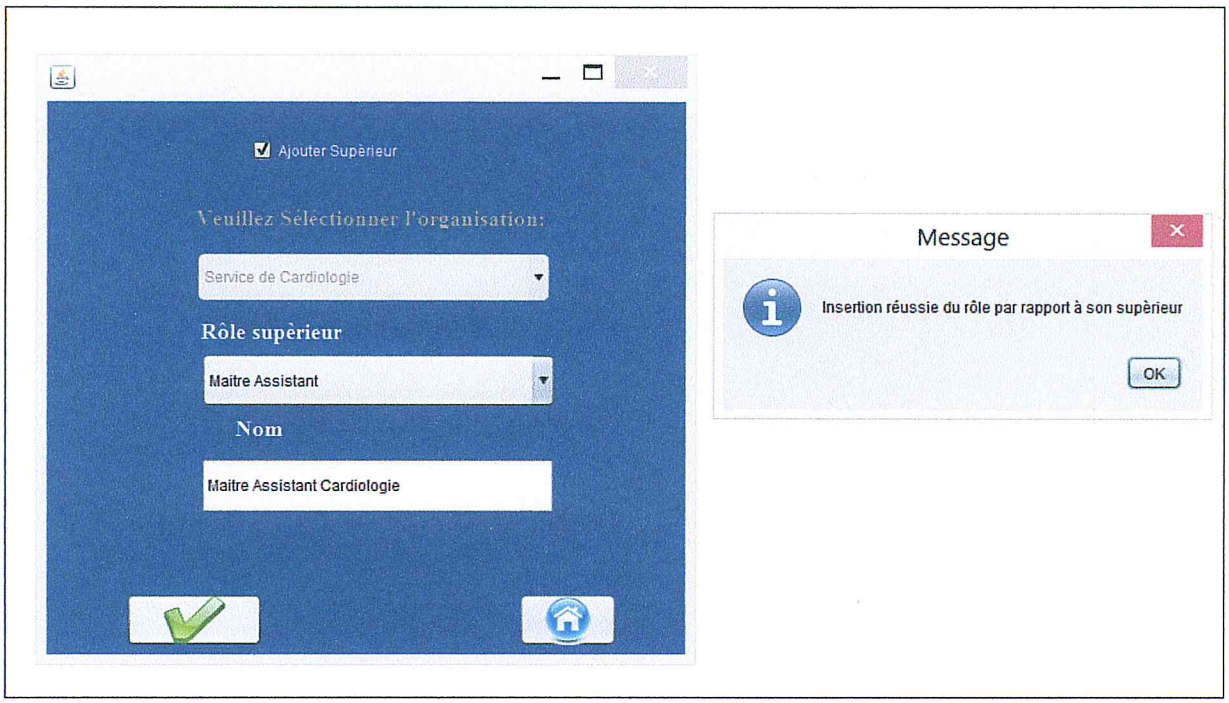

Figure 6.5 : ajout d'un rôle avec supérieur.

4.3. Modification d'une instance de concept :

Nous allons montrer comment se fait la modification de sujet (**Figure 6.6**) et de rôle(**Figure 6.7.1 et 6.7.2**) sachant que la modification d'organisation, d'action et d'objet se fait de la même manière que la modification de sujet mis à part la modification d'un mot de passe pour le sujet et la modification de vue et d'activité se fait de la même manière que la modification de rôle .

- Pour modifier un Sujet il faut aller dans le menu de l'interface principale (entités OrBAC) et cliquer sur modifier entité du menu déroulant et ainsi sélectionner ajouter sujet , pour modifier un sujet il faut modifier le nom et/ou le mot de passe.

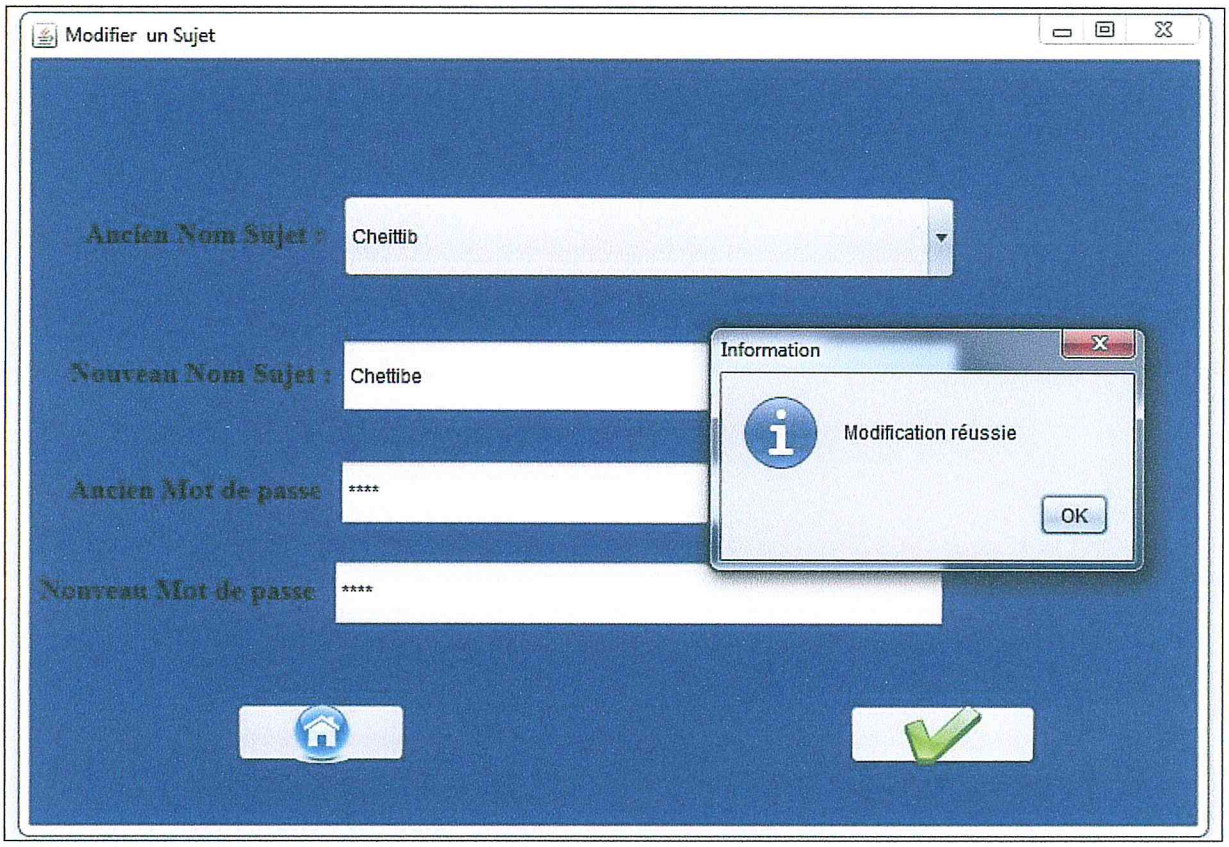

Figure 6.6: Modification d'un sujet.

- Pour modifier un rôle il faut aller dans le menu de l'interface principale (entités OrBAC) et cliquer sur modifier entité du menu déroulant et ainsi sélectionner modifier rôle. Quand un rôle a été modifié avec succès une boite de message apparait. **(Figure 6.7.3)**

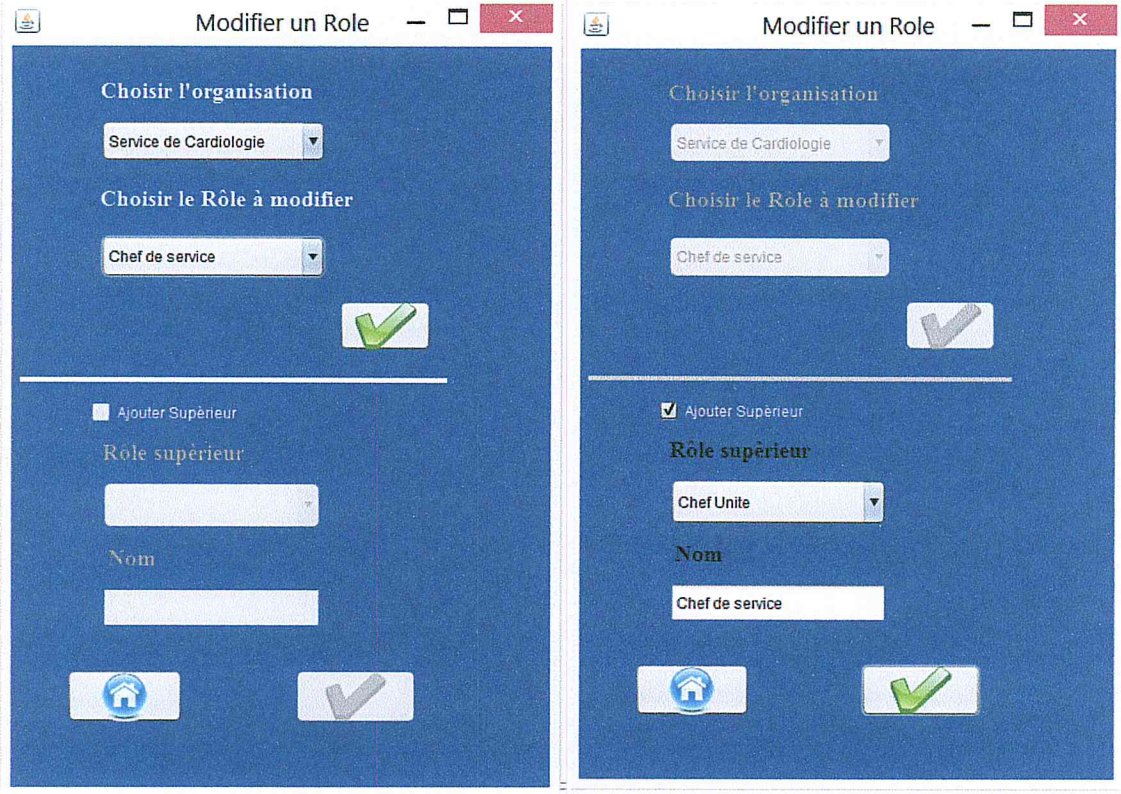

Figure 6.7.1 : Modification d'un rôle avec supérieur.

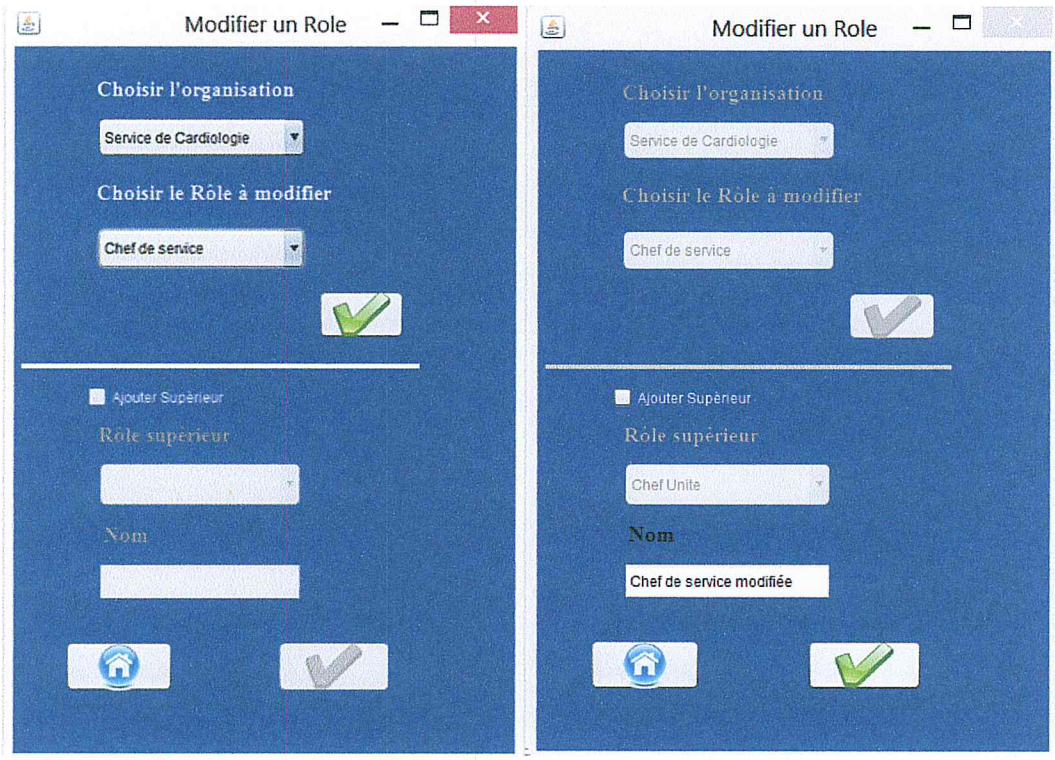

Figure 6.7.2: Modification d'un rôle sans supérieur.

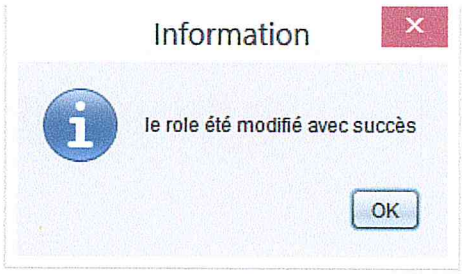

Figure 6.7.3 : Boite de message modification réussie dans les deux cas.

4.4. Suppression d'une instance de concept :

Nous allons montrer comment se fait la suppression d'objet (**Figure 6.8**) à la suppression d'objet on a une boite de message pour montrer la réussite de celle-ci (**Figure 6.9**) sachant que la suppression des autres concepts se fait de la même manière.

Remarque :

On ne peut pas supprimer les instances de concept qui rentre dans la description d'un autre concept.

- Pour Supprimer un objet il faut aller dans le menu de l'interface principale (entités OrBAC) et cliquer sur supprimer une entité du menu déroulant et ainsi sélectionner supprimer Objet .

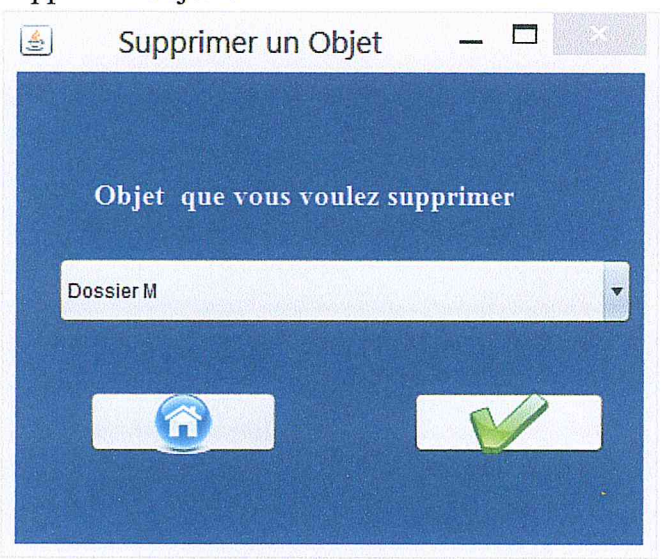

Figure 6.8 : Suppression d'objet .

Figure 6.9 : boite de message suppression d'objet.

4.5. Scénario montrant les fonctionnalités de l'application :

4.5.1. Ajout d'organisation :

Nous commençons par ajouter l'organisation « Service de Cardiologie ». **(Figure 6.10).**à l'ajout d'une organisation une boite de message apparait.

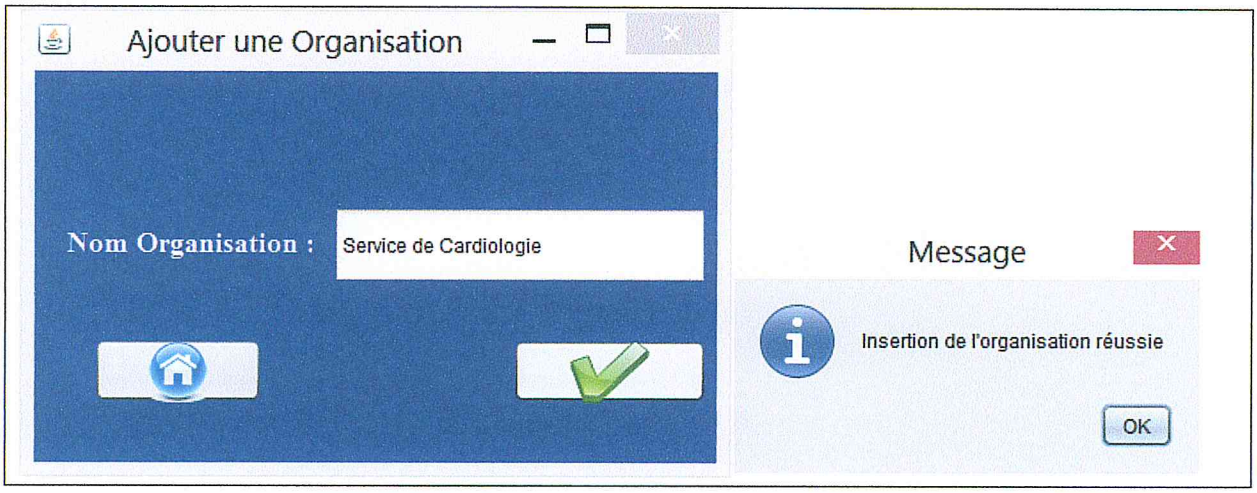

Figure 6.10 : Ajout d'une organisation.

4.5.2. Ajout rôle :

Nous ajoutons le rôle « Chef de Service » dans l'organisation « Service de Cardiologie ». **(Figure 6.11)**

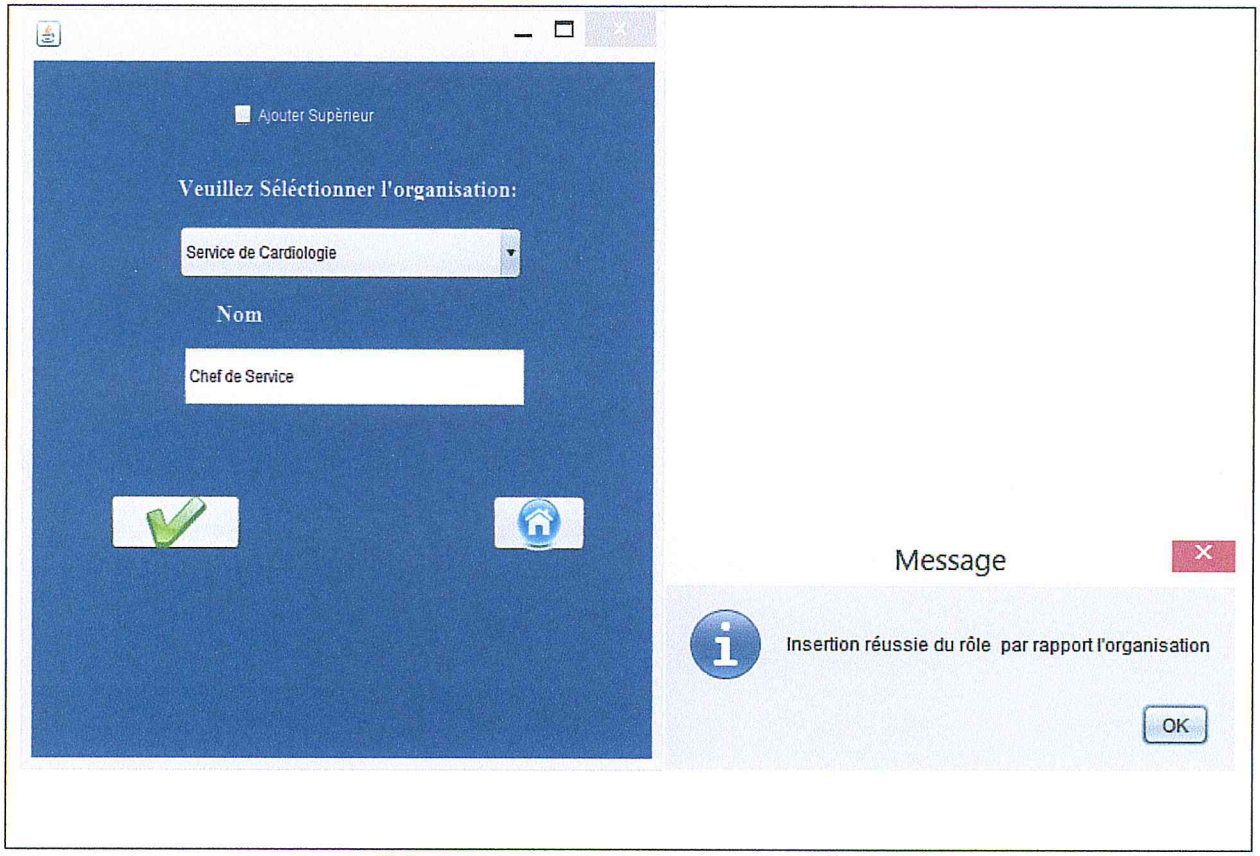

Figure 6.11 : Ajouter rôle dans une organisation.

4.5.3. Ajout Activité:

Nous ajoutons l'activité « Activité générale» dans l'organisation « Service de Cardiologie ». (**Figure 6.12**)

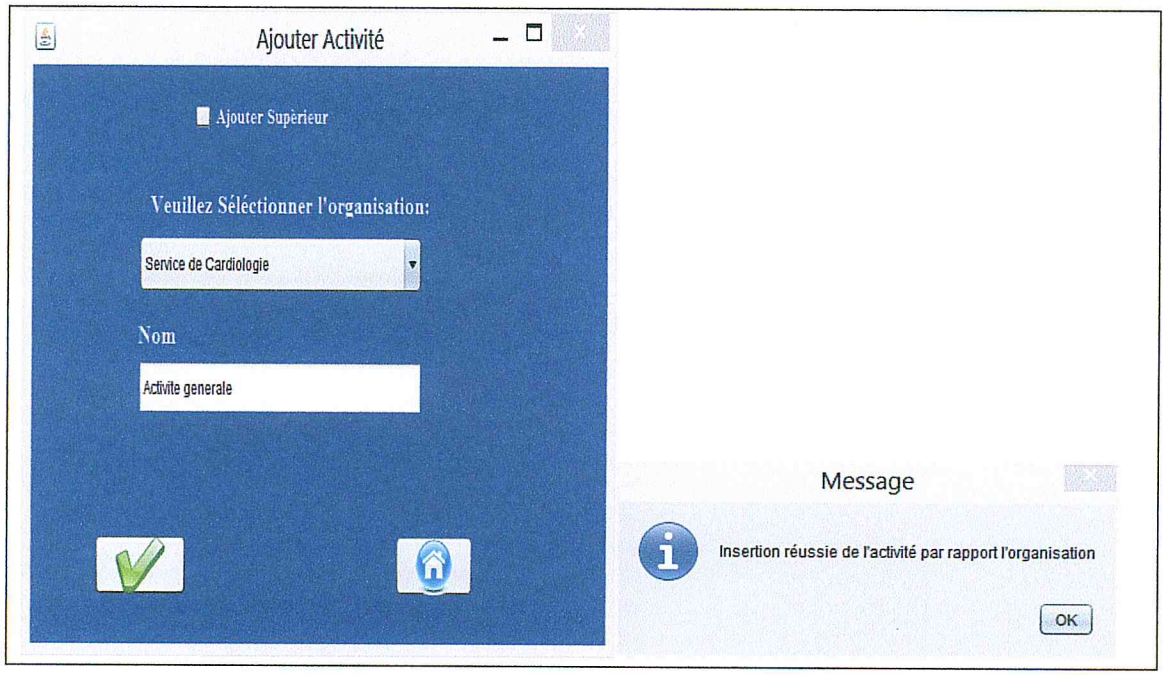

Figure 6.12 : Ajouter activité dans une organisation.

4.5.4. Ajout Vue:

Nous ajoutons la vue « Dossier Malade» dans l'organisation « Service de Cardiologie ». (**Figure 6.13**)

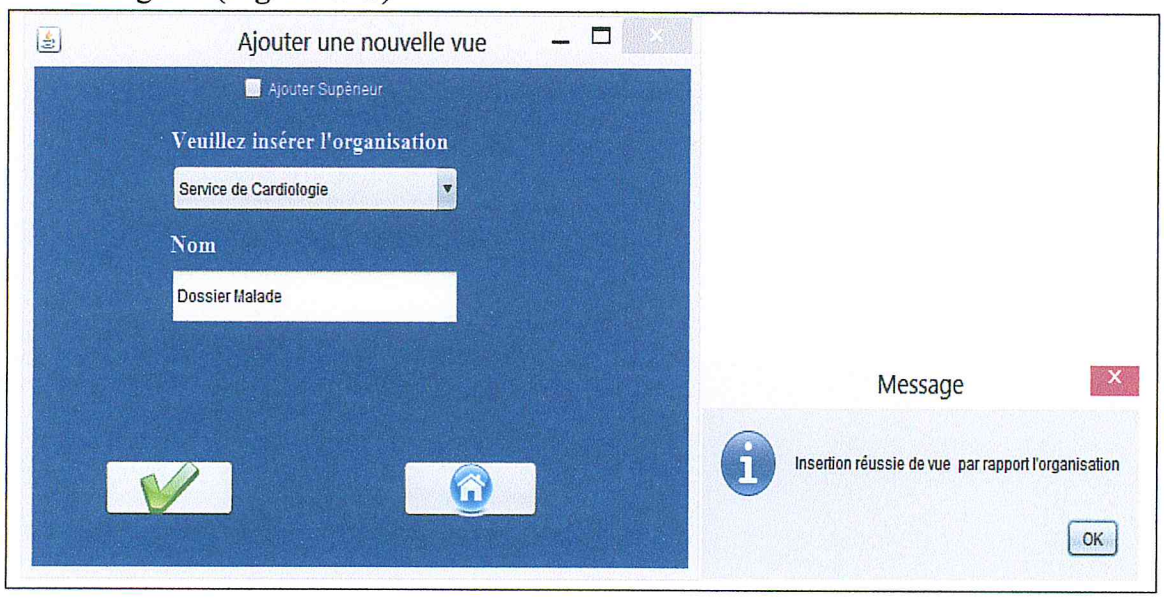

Figure 6.13 : Ajouter Vue dans l'organisation.

4.5.5. Ajout des sous vue:

Nous ajoutons les sous vues « dossier administratif » et « dossier médical » à la vue « dossier malade », puisque cela se fait de la même manière nous montrons que l'ajout de la sous vue « Dossier Administratif ». **(Figure 6.14)**

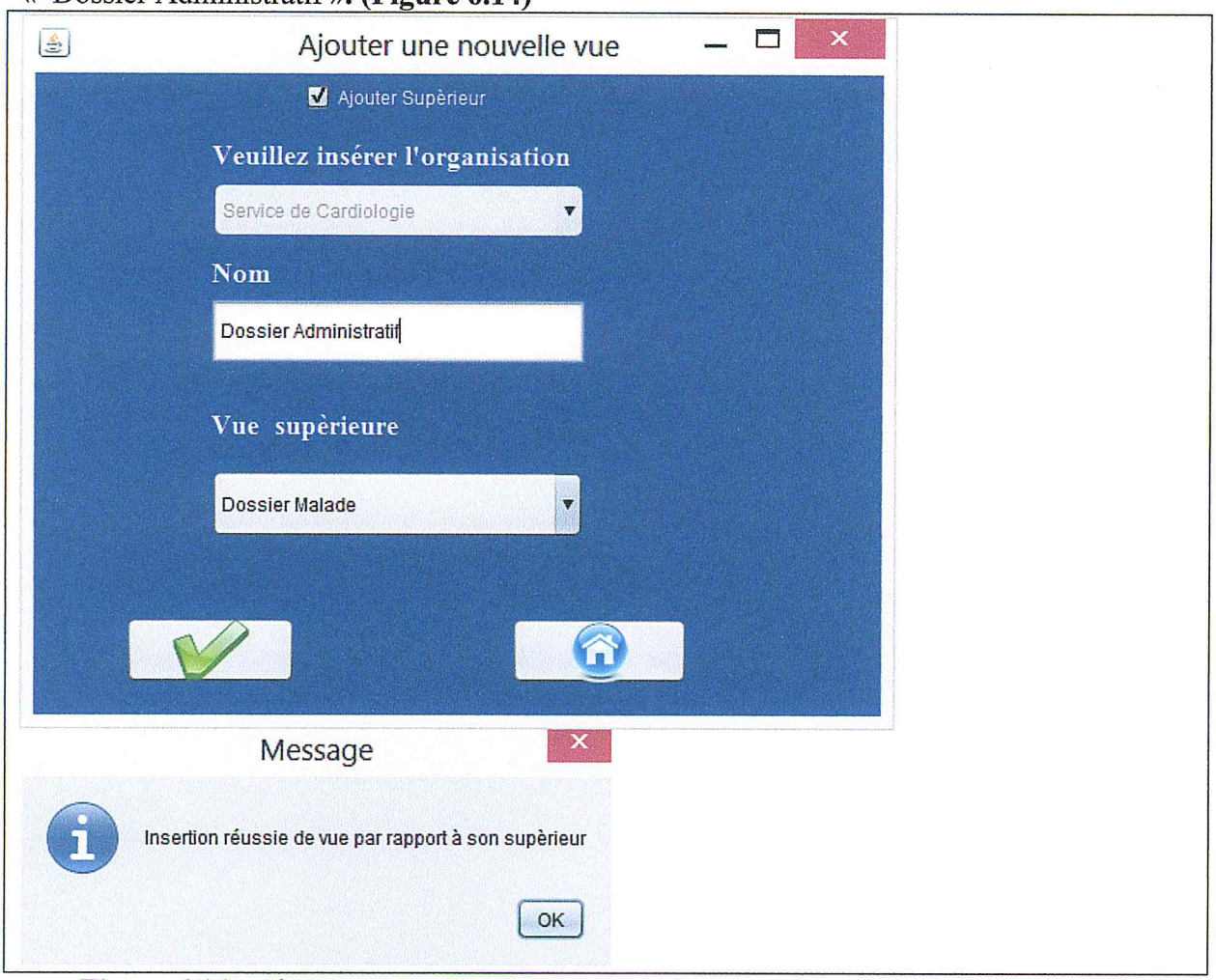

Figure 6.14 : Ajouter une sous vue par rapport à sa supérieure.

4.5.6. Ajout des sous activités:

Nous ajoutons la sous activité « gérer »à l'activité « Activité générale ». **(Figure 6.15)**

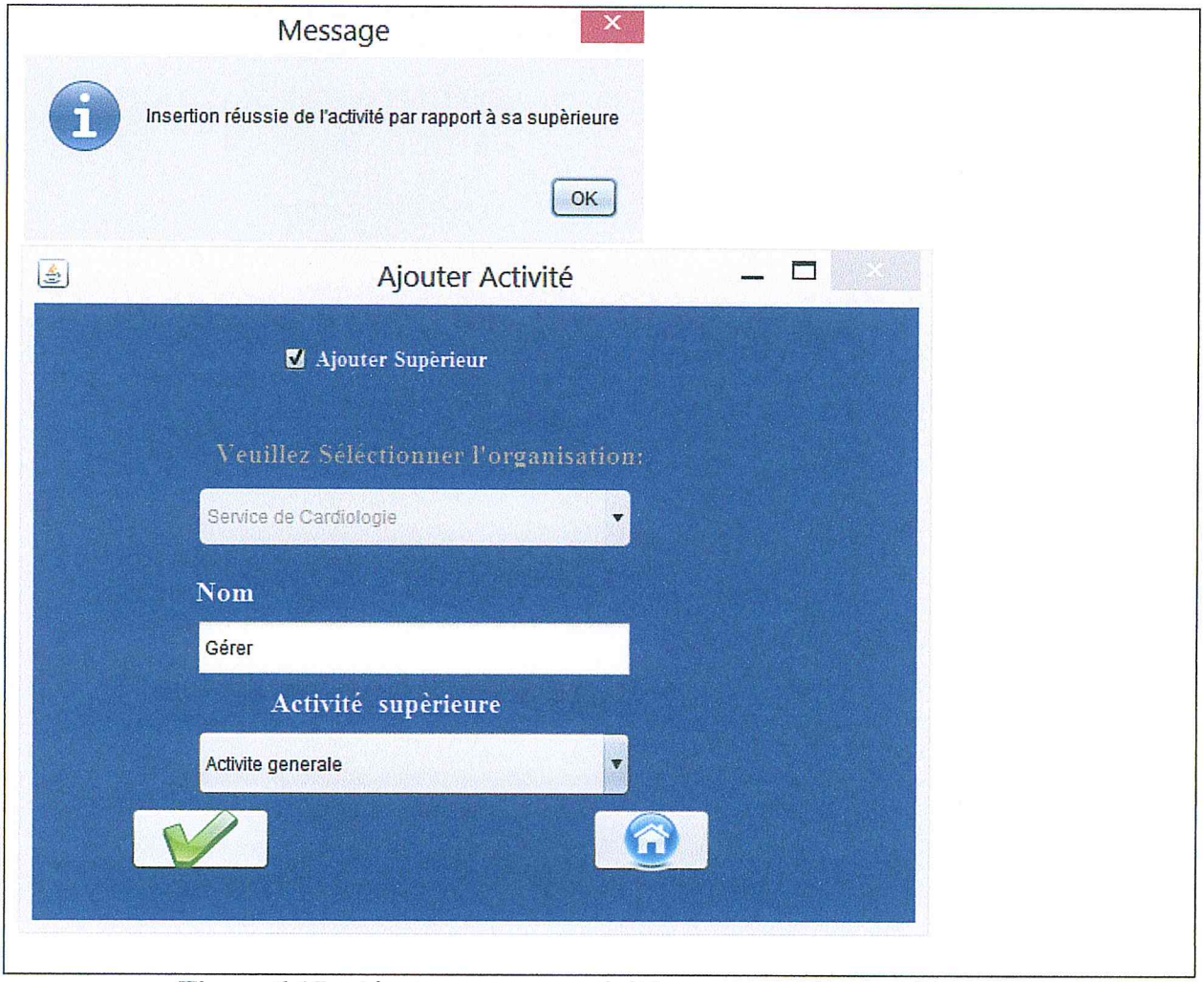

Figure 6.15 : Ajouter une sous activité par rapport à sa supérieure.

4.5.7. Ajout de sujet:

Nous ajoutons le sujet « Bouafia », à l'ajout d'un sujet on procède l'ajout d'un mot de passe dans l'optique d'ajouter ce sujet aux cessionnaires. **(Figure 6.16)**

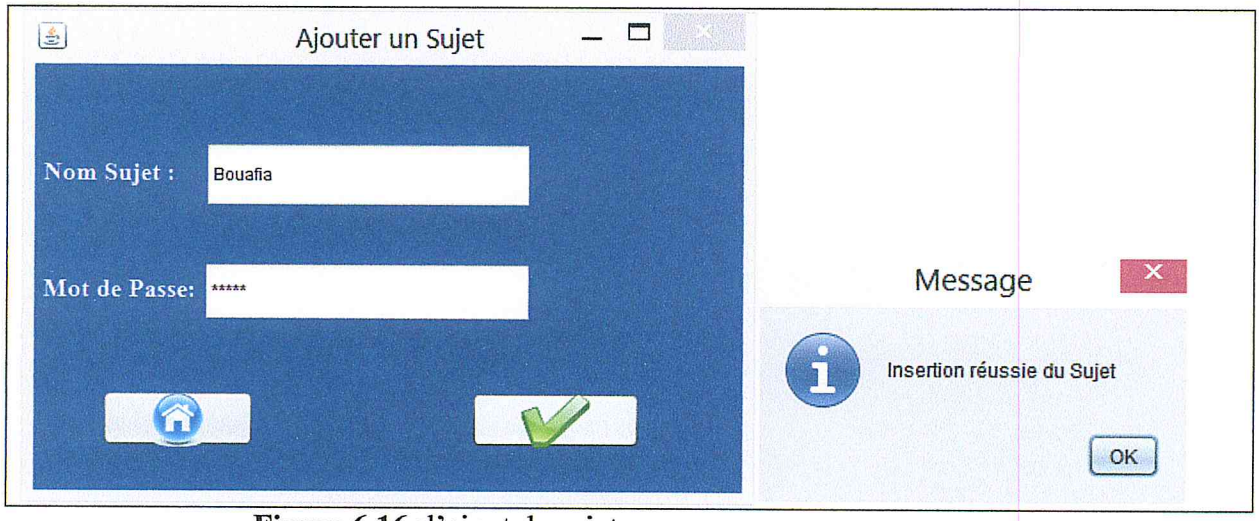

Figure 6.16 :l'ajout de sujet.

4.5.8. Ajout d'action:

Nous ajoutons l'action « Lire». **(Figure 6.17)**

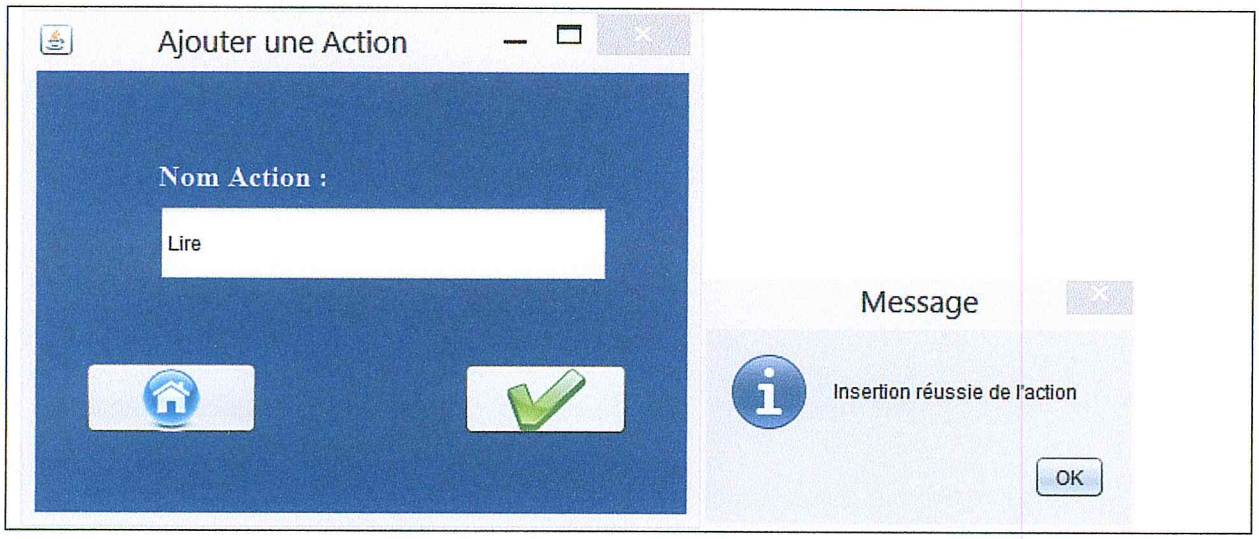

Figure 6.17 :l'ajout d'action.

4.5.9. Ajout d'objet:

Nous ajoutons les objets « Dossier M », « Dossier A », « fiche information », nous montrons que l'ajout de Dossier M puisque l'ajout se fait de la même manière. **(Figure 6.18)**

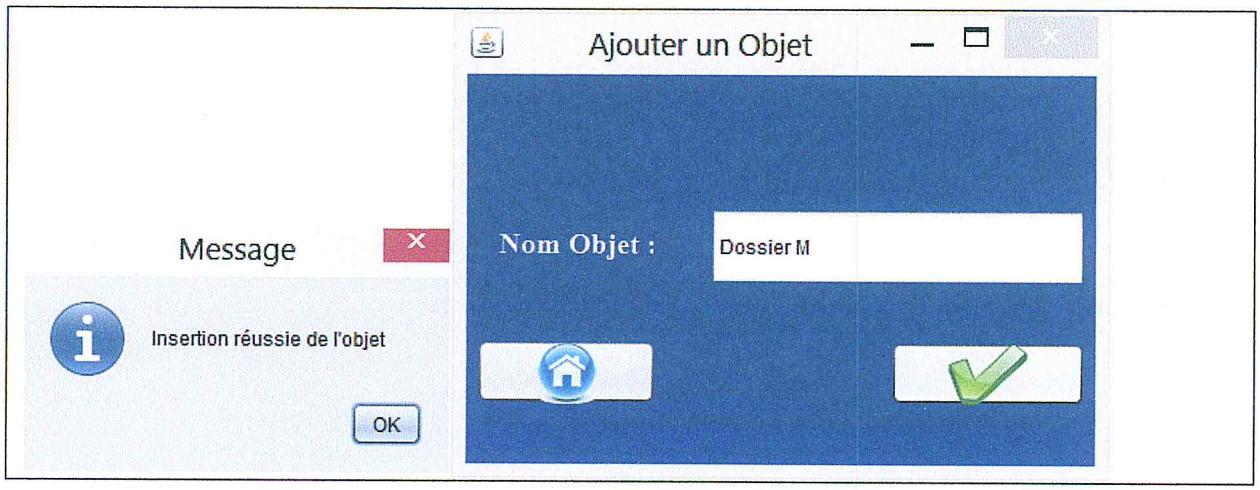

Figure 6.18 : Ajout d'objet .

4.5.10.Habilitation:

Nous habilitons le sujet « Bouafia » à jouer le rôle de Chef de service dans l'organisation Service de cardiologie suivant l'instance d'habilite (H1). (**Figure 6.19**)

- Habilite (H1)⊑ HabiliteS.Sujet(Bouafia) ⊓ HabiliteR.Role (Chef de Service) ⊓HabiliteOr.Organisation(Service de Cardiologie).

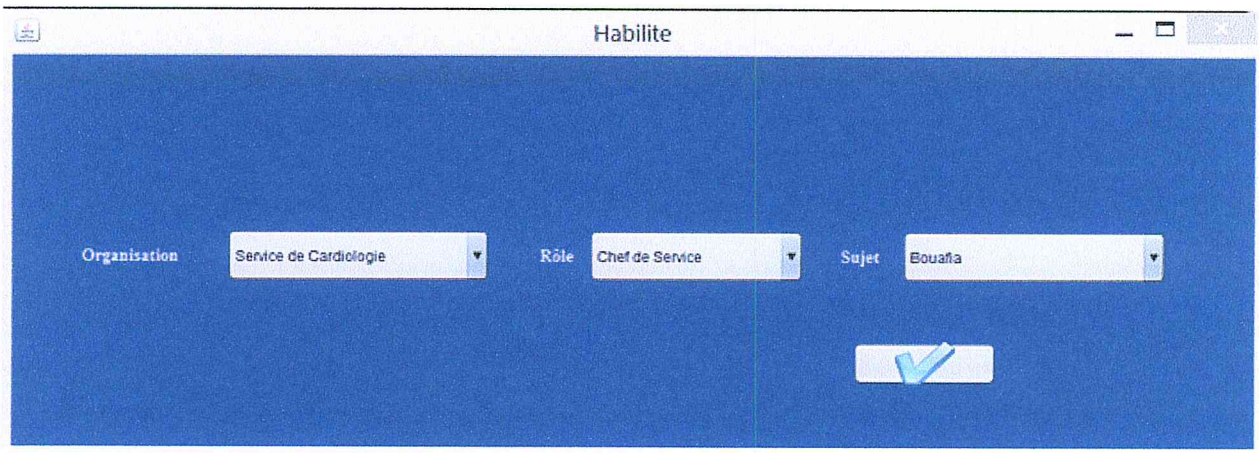

Figure 6.19 : Habiliter un Sujet à un rôle.

La (Figure 6.20) montre l'insertion de l'habilitation.

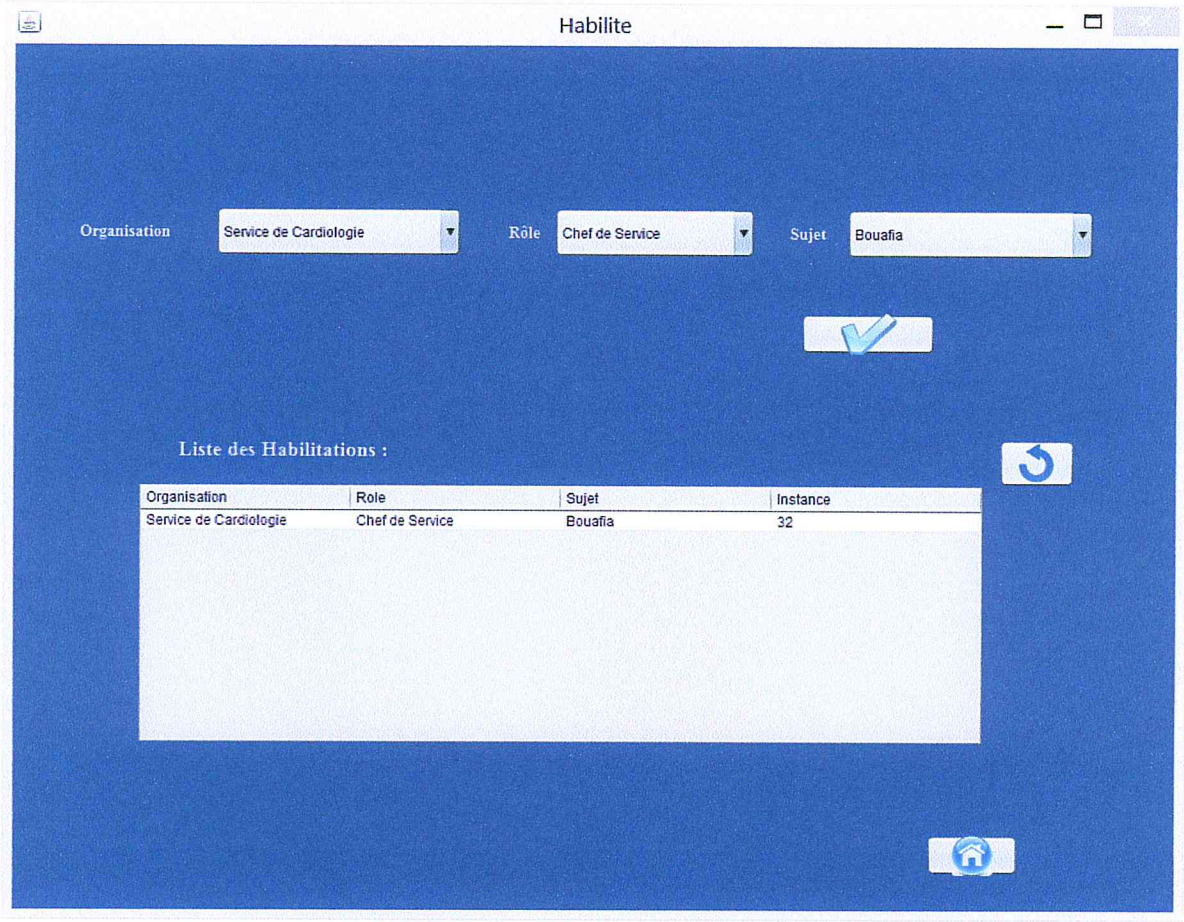

Figure 6.20 : Insertion de l'instance d'habilite.

4.5.11. Considération:

Nous considérons l'action « Lire » dans l'activité « activité générale » au sein de l'organisation Service de cardiologie suivant l'instance de considère (C1). (**Figure 6.21**)

- Considere (C1)⊑ ConsidereAc.Action (Lire) ⊓ ConsidereAv.Activite (Activité générale) ConsidereOr.Organisation (Service de cardiologie)

De la même manière nous considérons l'action créer dans l'activité gérer dans l'organisation service de cardiologie suivant l'instance (C2).

- Considere (C2)⊑ ConsidereAc.Action (Créer) ⊓ ConsidereAv.Activite (Gérer) ConsidereOr.Organisation (Service de cardiologie) ;

Figure 6.21 : Considérer une action dans une activité.

La (Figure 6.22) montre l'insertion de la considération.

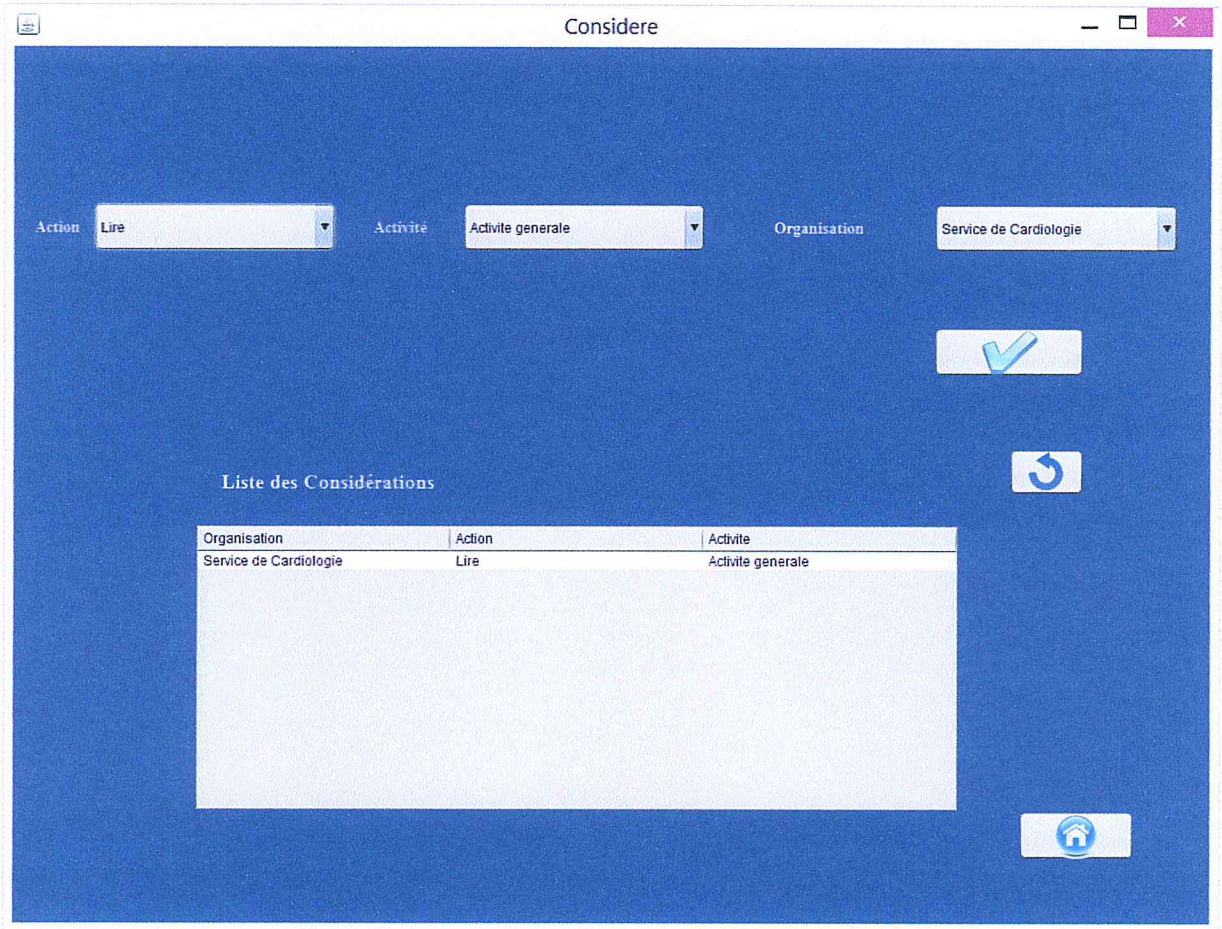

Figure 6.22 : Insertion d'une Considération.

4.5.12. Utilisation:

Nous utilisons l'objet« DossierM » dans la vue « Dossier Médical » au sein de l'organisation « Service de cardiologie » d'après l'instance de utilise (U1).

- Utilise (U1) ⊑ UtiliseO.Objet (DossierM) ⊓ UtiliseV.Vue (Dossier Médical) ⊓ UtiliseOr.Organisation (Service de Cardiologie) ;

Nous utilisons l'objet « DossierA » dans la vue « Dossier Administratif» au sein de l'organisation « Service de cardiologie » d'après l'instance de utilise (U2).

- Utilise (U2) ⊑ UtiliseO.Objet (DossierA) ⊓ UtiliseV.Vue (Dossier Administratif) ⊓ UtiliseOr.Organisation (Service de Cardiologie) ;

Nous utilisons l'objet « fiche information » dans la vue « Dossier Malade» au sein de l'organisation « Service de cardiologie » d'après l'instance de utilise (U3).(**Figure 6.23**)

- Utilise (U3) ⊑ UtiliseO.Objet (Fiche information) ⊓ UtiliseV.Vue (Dossier Malade) ⊓ UtiliseOr.Organisation (Service de Cardiologie) ;

Figure 6.23 : Utiliser un objet dans une vue.

La (Figure 6.24) montre l'insertion de l'utilisation.

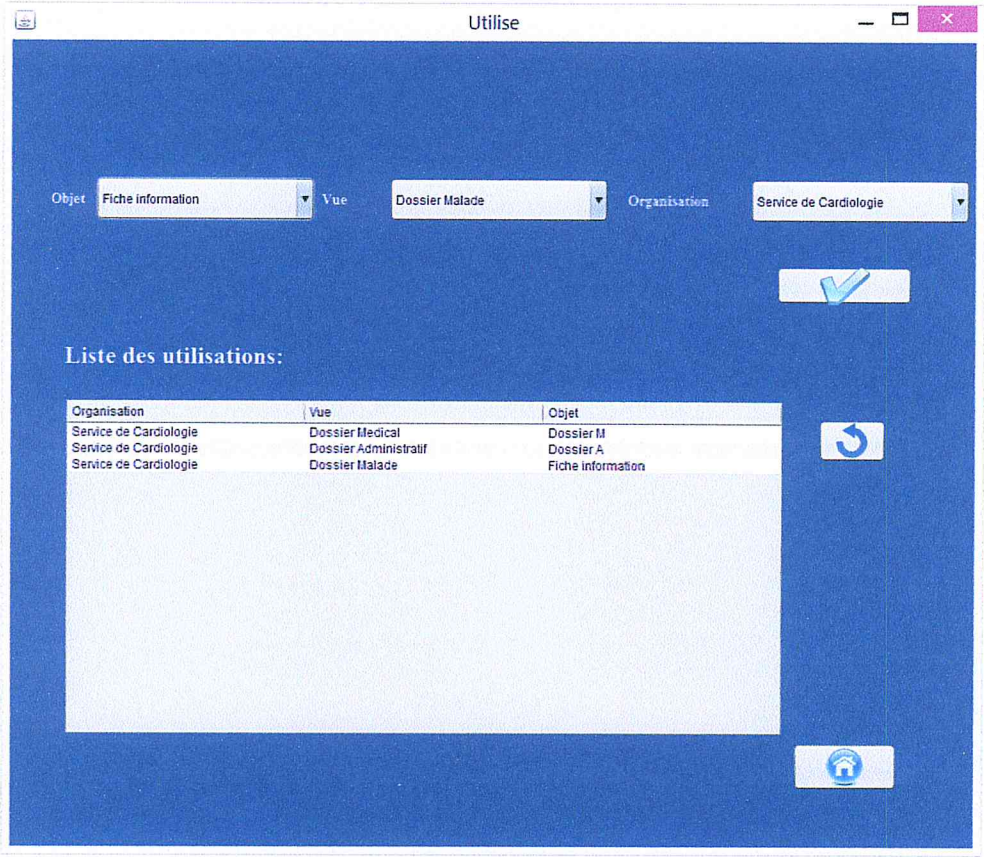

Figure 6.24 : Insertion de l'utilisation.

4.5.13. Permission Abstraite dans un contexte défaut (δ):

Nous insérons une nouvelle permission abstraite au sein de l'organisation **(δP1). (Figure 6.25).**

- δP1⊑ Chef de Service ⊓ Activité générale ⊓ Dossier Malade ⊓ Service de Cardiologie.

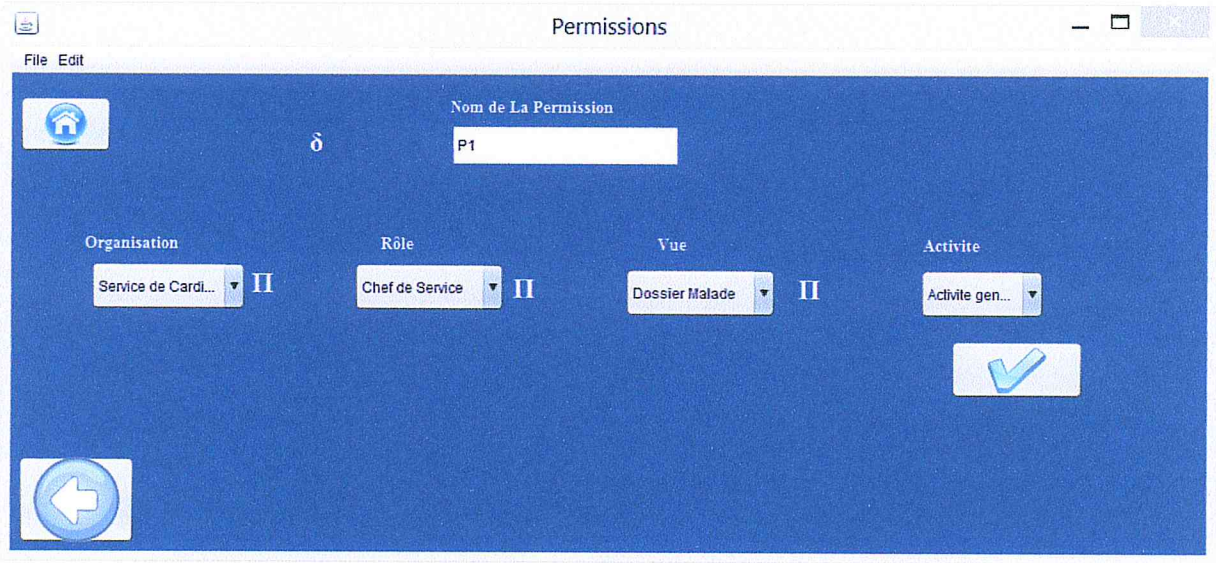

Figure 6.25.1 : Attribution d'une Permission.

A l'insertion d'une permission abstraite une boite de message apparait pour montrer que la permission a été insérée avec succès

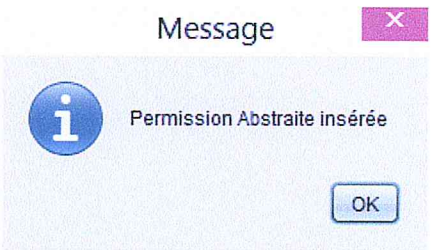

Figure 6.25.2 : Boite de message Permission insérée.

Dans le service de cardiologie, le sujet « Bouafia »est habilité à faire le rôle « Chef de service » (H1), l'objet « fiche information » est utilisé dans la vue « Dossier Malade » (U3), et l'action « Lire »est considérée dans « activité générale » (C1).

Donc la permission concrète suivante sera générée :

- EP1 ⊑ Bouafia ⊓ lire ⊓ fiche information.

Pour montrer que la Permission concrète EP1 a bien été générée nous procédons à l'interrogation du système.

Dans le service de Cardiologie , nous avons également une hiérarchie , dans la vue « Dossier Malade »nous avons les sous vues « dossier administratif » et « dossier médical » donc aura un héritage dans la hiérarchie de vues, dans l'activité « activité générale » nous avons la sous activité « gérer » donc on aura également un héritage dans la hiérarchie d'activités.
Cet héritage génèrera les permissions abstraites suivantes :

- δP1⊑ Chef de Service ⊓ Activité générale ⊓ Dossier Médical ⊓ Service de Cardiologie.
- δP1⊑ Chef de Service ⊓ Activité générale ⊓ Dossier Administratif ⊓ Service de Cardiologie.
- δP1⊑ Chef de Service ⊓ Gérer ⊓ Dossier Médical⊓ Service de Cardiologie.
- δP1⊑ Chef de Service ⊓ Gérer ⊓ Dossier Malade ⊓ Service de Cardiologie.
- δP1⊑ Chef de Service ⊓ Gérer⊓ Dossier Administratif ⊓ Service de Cardiologie.

Dans le service de cardiologie, l'objet « Dossier M » est utilisé dans la vue « Dossier Médical » (U1), l'objet « Dossier A » est utilisé dans la vue « Dossier Administratif» (U2), et l'action « créer »est considérée dans l'activité « gérer » (C2).

Donc les permissions concrètes suivantes seront générées suivant les permissions abstraites héritées :

- EP2⊑ Bouafia ⊓ Lire⊓ Dossier M.

Pour montrer que la Permission concrète EP2 a bien été générée nous procédons à l'interrogation du système

- EP3⊑ Bouafia ⊓ Lire ⊓ Dossier A.

Pour montrer que la Permission concrète EP3 a bien été générée nous procédons à l'interrogation du système

- EP4⊑ Bouafia ⊓ Créer⊓ Dossier M.

Pour montrer que la Permission concrète EP4 a bien été générée nous procédons à l'interrogation du système

- EP5⊑ Bouafia ⊓ Créer ⊓ Fiche information.

Pour montrer que la Permission concrète EP5 a bien été générée nous procédons à l'interrogation du système

- EP6⊑ Bouafia ⊓ Créer ⊓ Dossier A.

Pour montrer que la Permission concrète EP6 a bien été générée nous procédons à l'interrogation du système (**Figure 6.26**)

Nous montrons l'interrogation du système pour la permission EP6, en sachant que cette interrogation se fait de la même manière pour les permissions concrètes précédentes et qu'elle est accordée dans chaque cas. (**Figure 6.26**)

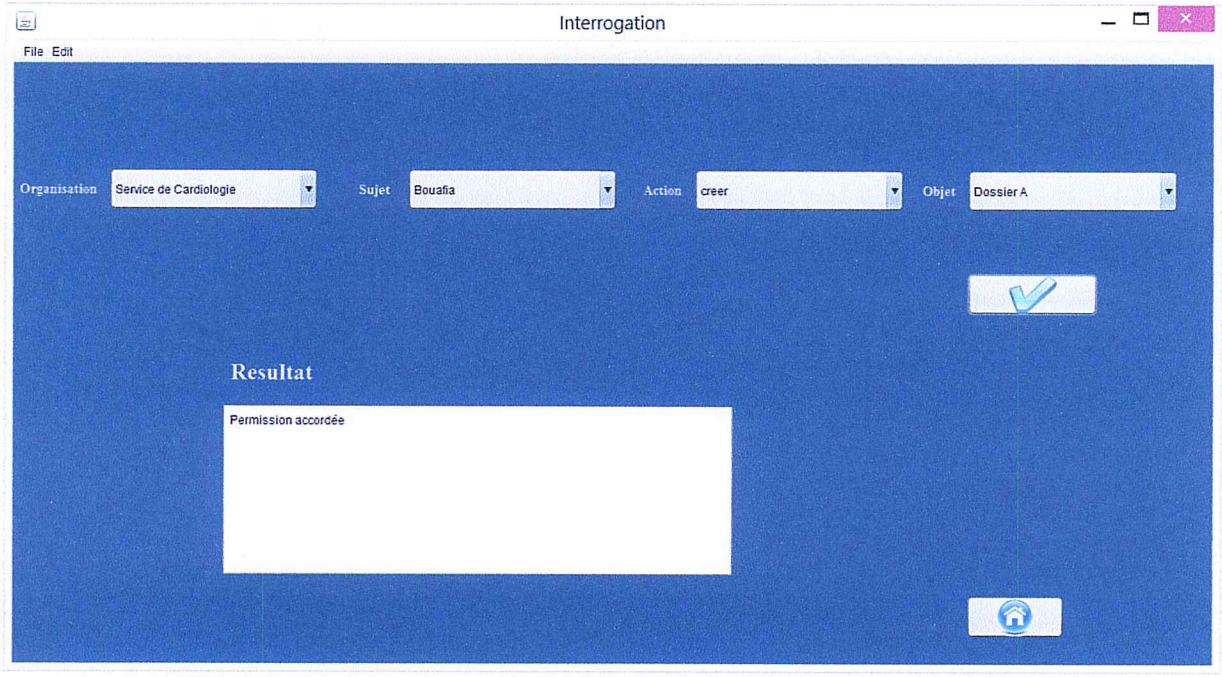

Figure 6.26 : Interrogation du système pour montrer l'héritage dans la hiérarchie.

4.5.14. Délégation :

Dans cette partie nous allons montrer comment se fait la délégation dans notre système, la délégation étant le but principal de notre étude nous allons détailler cette partie, nous allons exprimer les différentes délégations en utilisant les entités, relations et permissions créées dans les sections précédentes du scénario.

4.5.14.1. L'ajout de cessionnaires:

Pour qu'un sujet de l'organisation puisse déléguer ses droits il faut tout d'abord que l'administrateur lui en donne le droit, pour cela l'administrateur l'ajoute aux cessionnaires .Il peut aussi le supprimer des cessionnaires si ce sujet n'a plus le droit de déléguer ses droits.

Nous allons ajouter le sujet « Bouafia » aux cessionnaires pour montrer l'expression de la délégation avec ses différents types.**(Figure 6.27)**

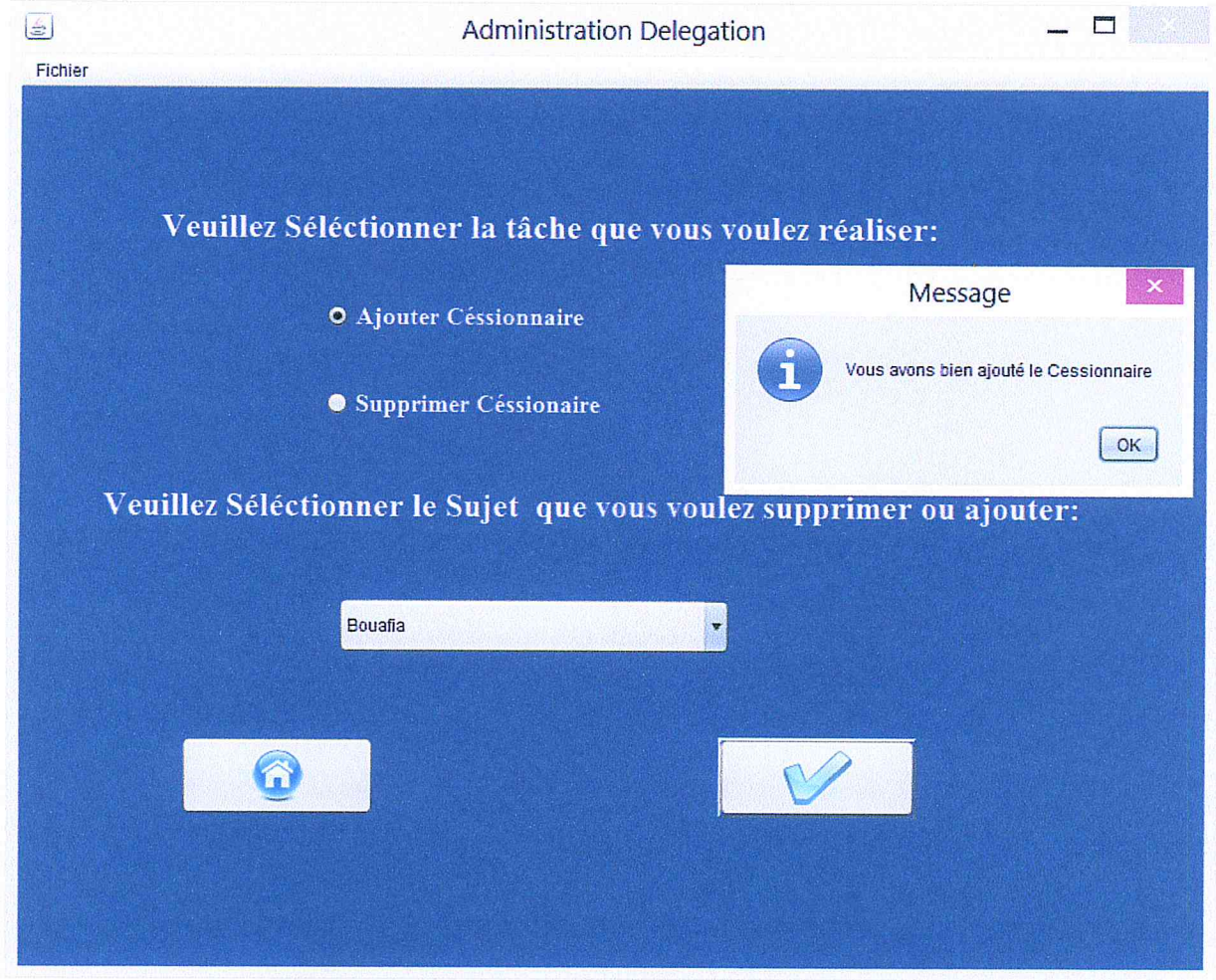

Figure 6.27 : Ajout de cessionnaire.

Pour continuer notre scénario, nous procédons à l'ajout du rôle « Chef d'unité », l'ajout du sujet « Boureghda », et on finit par l'habilitation de « Boureghda » au rôle « chef d'unité » toujours dans l'organisation « service de cardiologie »suivant l'instance d'habilite (H2).

4.5.14.2. Délégation de Permission de rôle:

On procède à la délégation de permission de rôle qui est une délégation totale, le rôle chef de service délègue ses permissions au rôle chef d'unité suivant l'instance rôle délégation **PRD.** **(Figure 6.29)**

Avant cette délégation le cessionnaire doit s'authentifier pour lui donner la main de déléguer que le rôle qu'il obtient et pas un autre, et aussi pour vérifier que ce sujet est bien cessionnaire.

(Figure 6.28)

Figure 6.28 : authentification du cessionnaire.

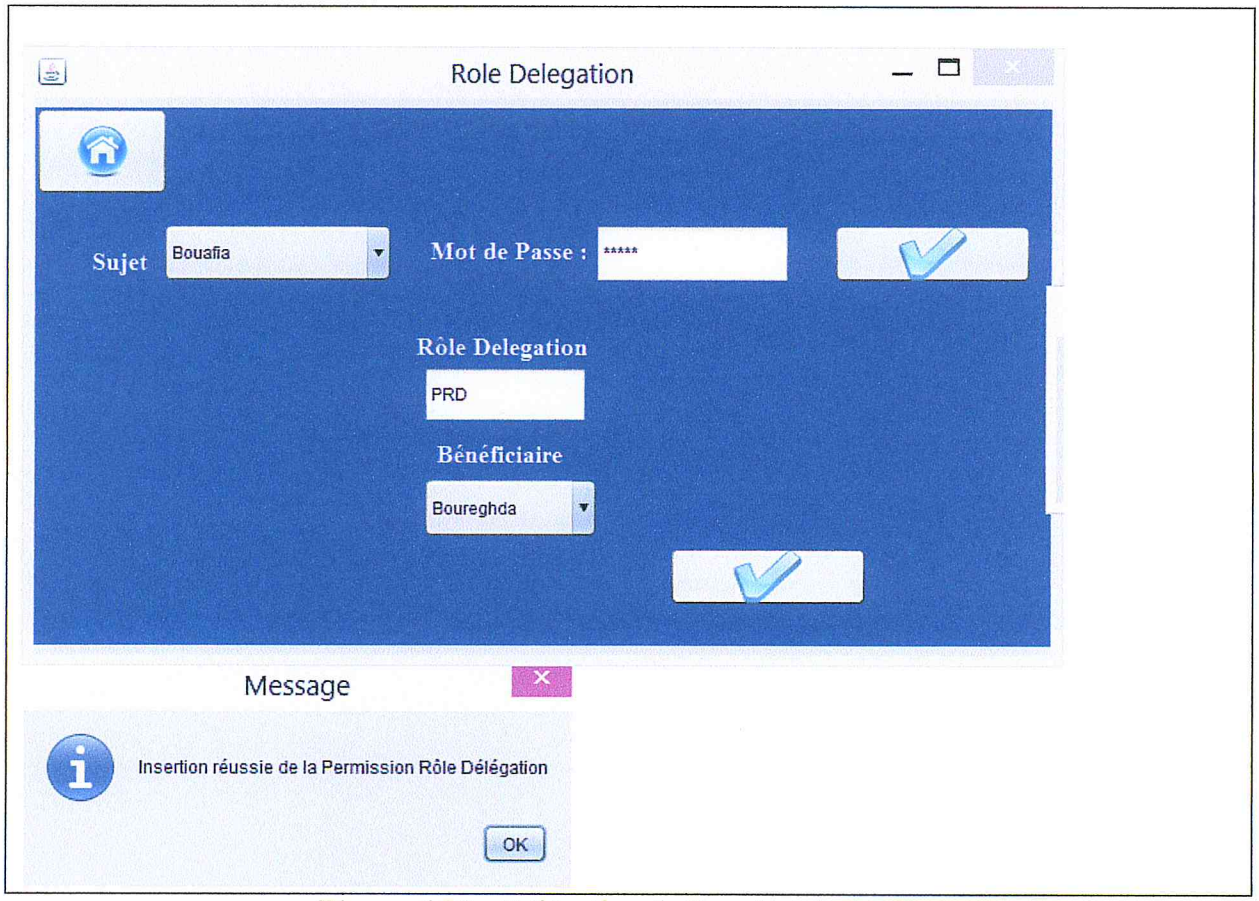

Figure 6.29 : Délégation de Permission de rôle.

- Cette délégation de permission de rôle va générer une nouvelle permission abstraite(δP2), cette même permission abstraite va générer des permissions héritées, et des permissions concrètes seront générées de la même manière que dans la section (**4.5.13**)
- La permission abstraite :
  δP2⊑ Chef d'unité ⊓ Activité générale ⊓ Dossier Malade ⊓ Service de Cardiologie.

- Donc la permission concrète suivante sera générée :
  EP7 ⊑ Boureghda⊓ lire ⊓ fiche information.
  Pour montrer que la Permission concrète EP7 a bien été générée nous procédons à l'interrogation du système.
- Les permissions héritées :

- δP2⊑ Chef d'unité ⊓ Activité générale ⊓ Dossier Médical ⊓ Service de Cardiologie.
- δP2⊑ Chef d'unité ⊓ Activité générale ⊓ Dossier Administratif ⊓ Service de Cardiologie.
- δP2⊑ Chef d'unité ⊓ Gérer ⊓ Dossier Médical⊓ Service de Cardiologie.
- δP2⊑ Chef d'unité ⊓ Gérer ⊓ Dossier Malade ⊓ Service de Cardiologie.
- δP2 ⊑ Chef d'unité ⊓ Gérer⊓ Dossier Administratif ⊓ Service de Cardiologie.

Donc les permissions concrètes suivantes seront générées suivant les permissions abstraites héritées :

- EP8⊑ Boureghda⊓ ⊓ Lire⊓ Dossier M.

Pour montrer que la Permission concrète EP8 a bien été générée nous procédons à l'interrogation du système

- EP9⊑ Boureghda⊓ ⊓ Lire ⊓ Dossier A.

Pour montrer que la Permission concrète EP9 a bien été générée nous procédons à l'interrogation du système

- EP10⊑ Boureghda⊓ ⊓ Créer⊓ Dossier M.

Pour montrer que la Permission concrète EP10 a bien été générée nous procédons à l'interrogation du système

- EP11⊑ Boureghda⊓ ⊓ Créer ⊓ Fiche information.

Pour montrer que la Permission concrète EP11 a bien été générée nous procédons à l'interrogation du système

- EP12⊑ Boureghda⊓ ⊓ Créer ⊓ Dossier A.

Pour montrer que la Permission concrète EP12 a bien été générée nous procédons à l'interrogation du système (**Figure 6.30**)

Nous montrons l'interrogation du système pour la permission EP12, en sachant que cette interrogation se fait de la même manière pour les permissions concrètes précédentes et qu'elle est accordée dans chaque cas. **(Figure 6.30)**

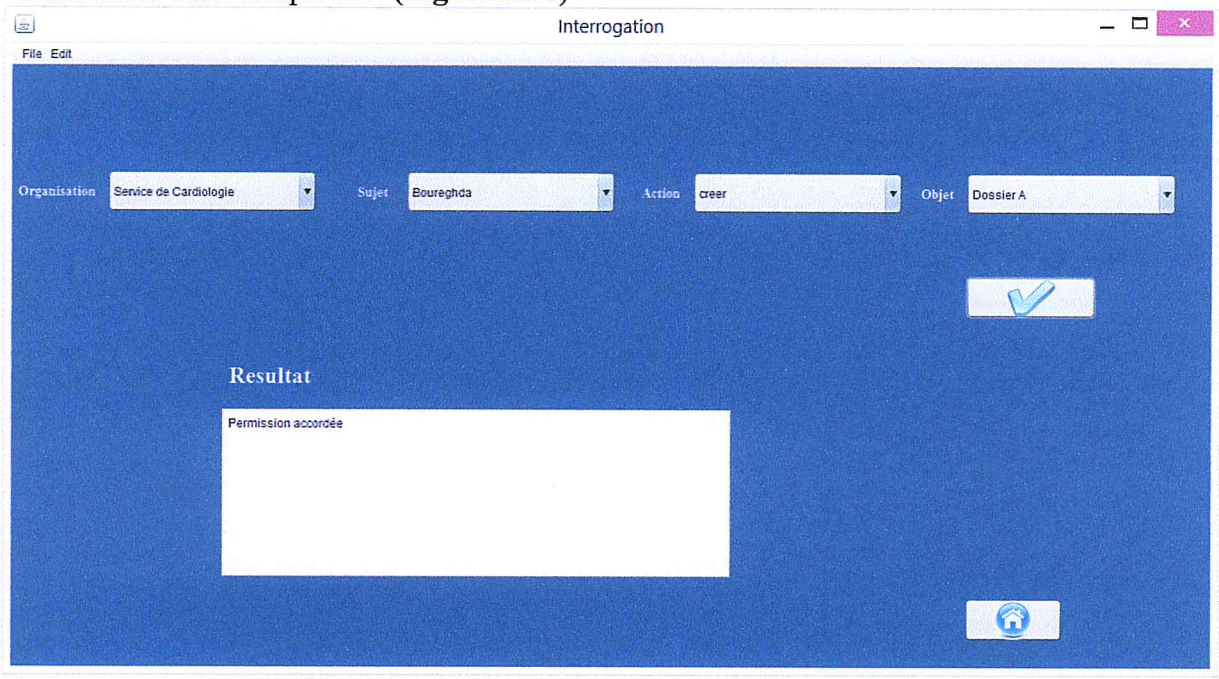

Figure 6.30 : Interrogation du système pour montrer l'héritage dans la hiérarchie après une délégation de permission de rôle.

4.5.14.3. Révocation GD:

Pour la révocation GD (grant-dependant) , c'est-à-dire que seul le cessionnaire initial de la délégation peut révoquer les droits donnés par celle-ci , nous allons prendre l'exemple de la délégation de permission de rôle.

Pour révoquer PRD , le cessionnaire (Bouafia) doit s'authentifier de la même manière que dans la **(Figure 6.28).**ensuite il procède à la révocation de la licence qu'il souhaite **(Figure 6.31).**

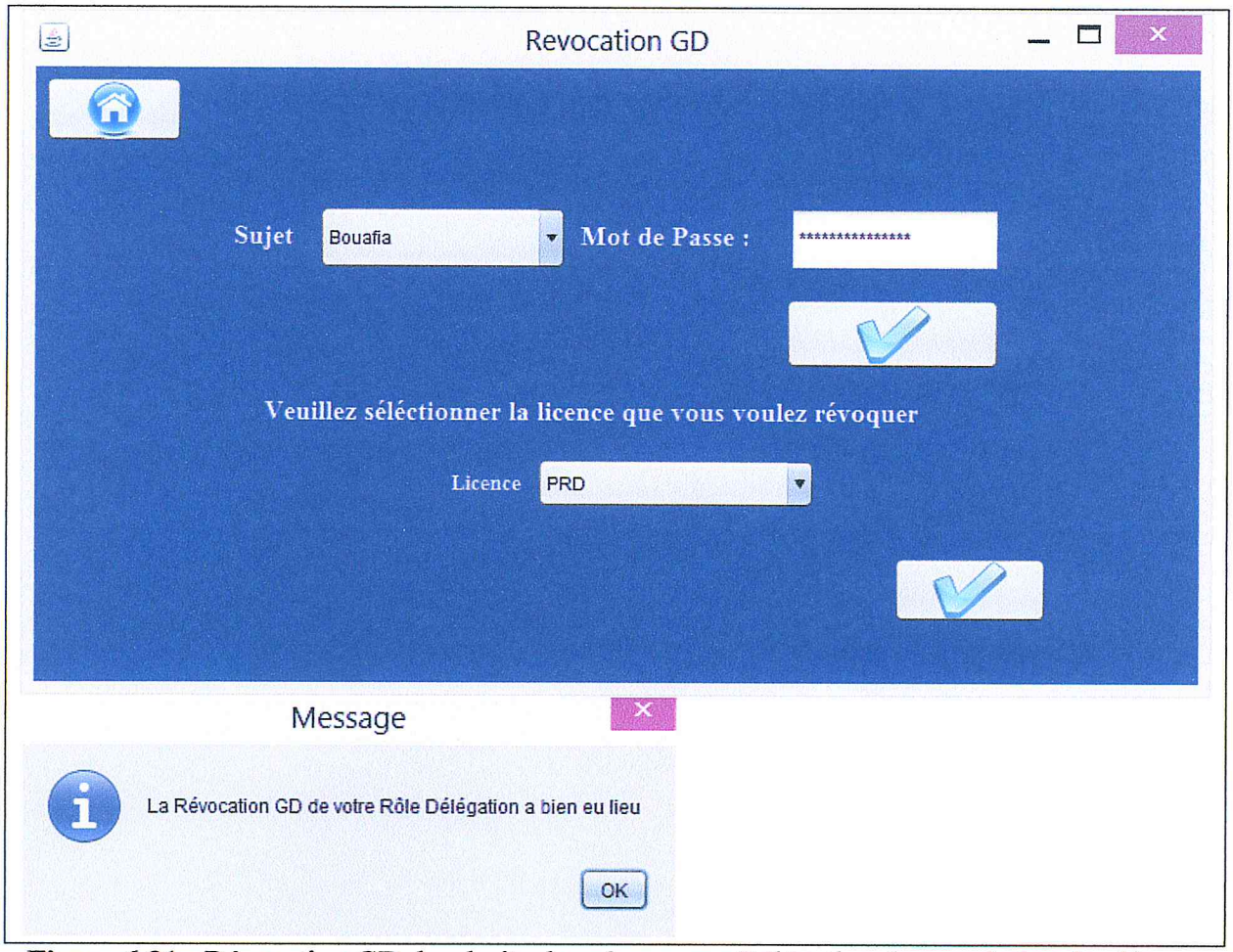

Figure 6.31 : Révocation GD des droits donnés par une Délégation de permission de rôle.

Pour montrer que cette révocation a bien eu lieu on procède à l'interrogation du système dans tous les cas de permissions concrètes. la **Figure 6.32** montre l'interrogation du système dans le même cadre que la **Figure 6.30,** nous remarquons que la permission n'est plus accordée.

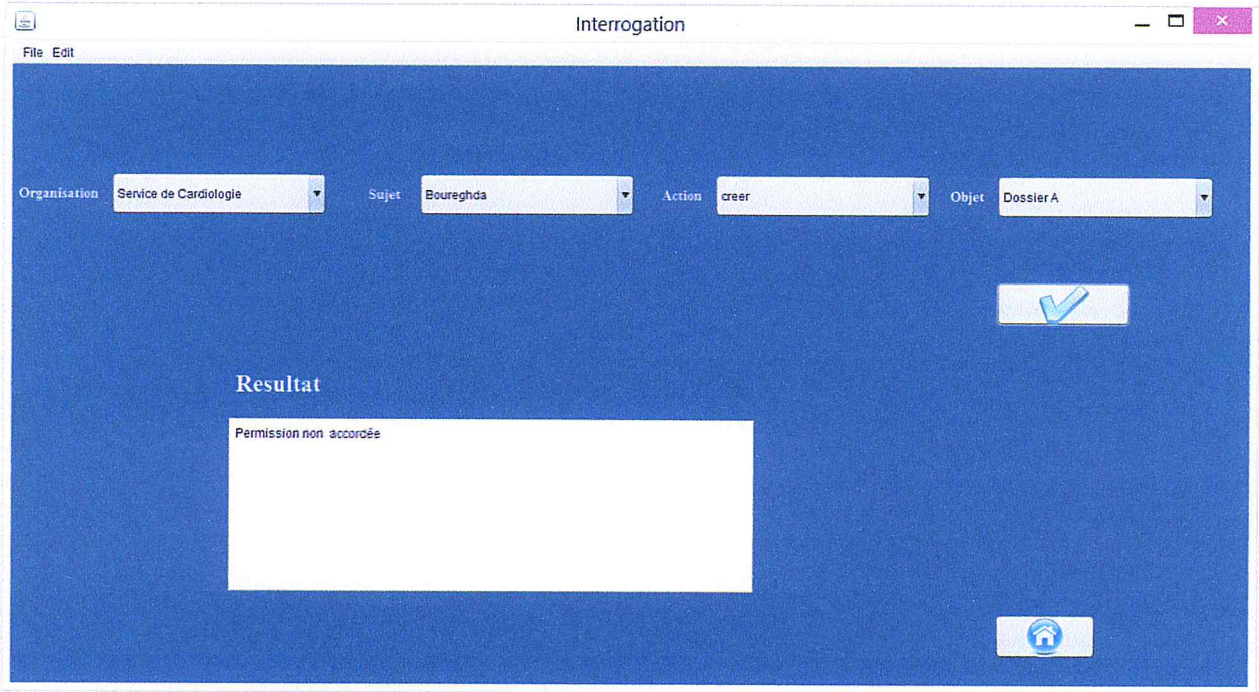

Figure 6.32 : Interrogation après révocation GD.

4.5.14.4. Délégation de Permission de Licence :

4.1. Délégation de Permission de Licence dans le contexte défaut :
On procède à la délégation de permission de Licence qui est une délégation partielle dans un contexte défaut qui est « Vacances », le sujet « Bouafia » délègue une de ses permissions qui est la permission de lire l'objet dossier M au sujet « Boureghda » suivant l'instance permission délégation **PLD (Figure 6.34).**
Avant cette délégation le cessionnaire doit s'authentifier pour lui donner la main de déléguer que les permissions qu'il obtient et pas d'autres, et aussi pour vérifier que ce sujet est bien cessionnaire. **(Figure 6.33)**

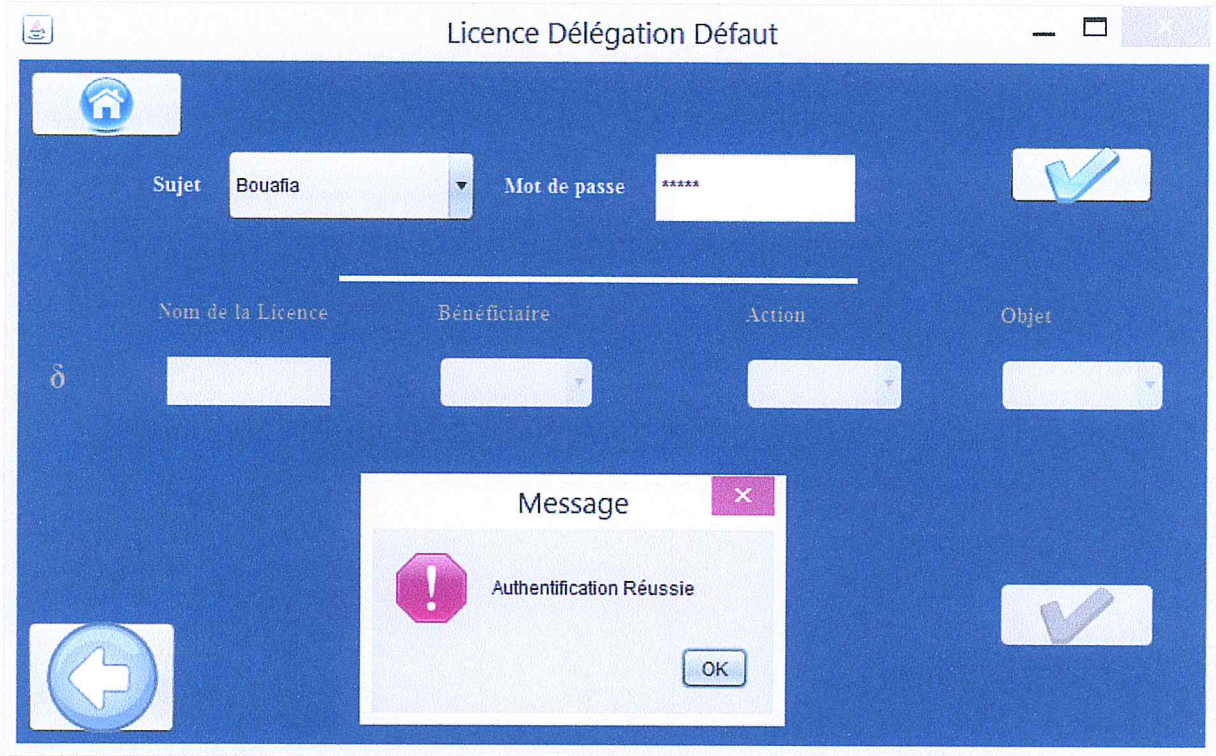

Figure 6.33 : authentification du cessionnaire pour licence délégation.

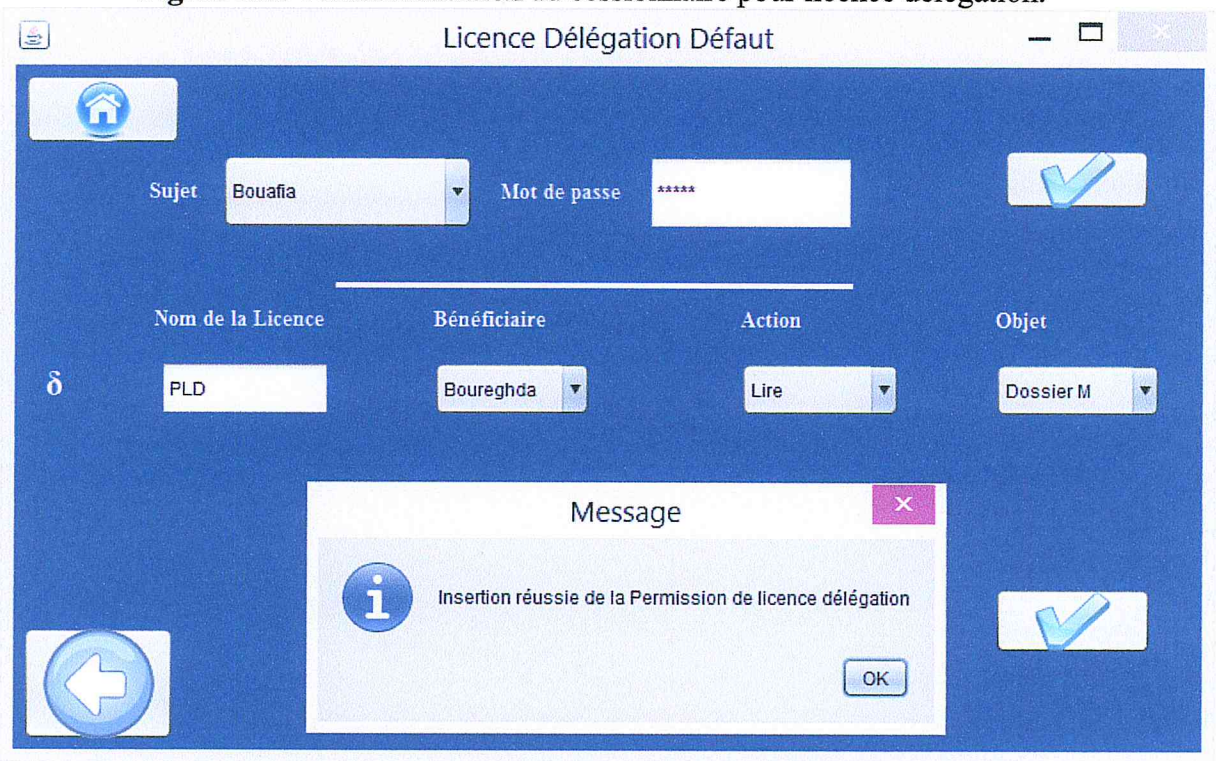

Figure 6.34 : Délégation de permission de licence dans le contexte défaut.

Pour montrer que la délégation de permission de licence délégation est réussie on procède à l'interrogation du système. (**Figure 6.35**)

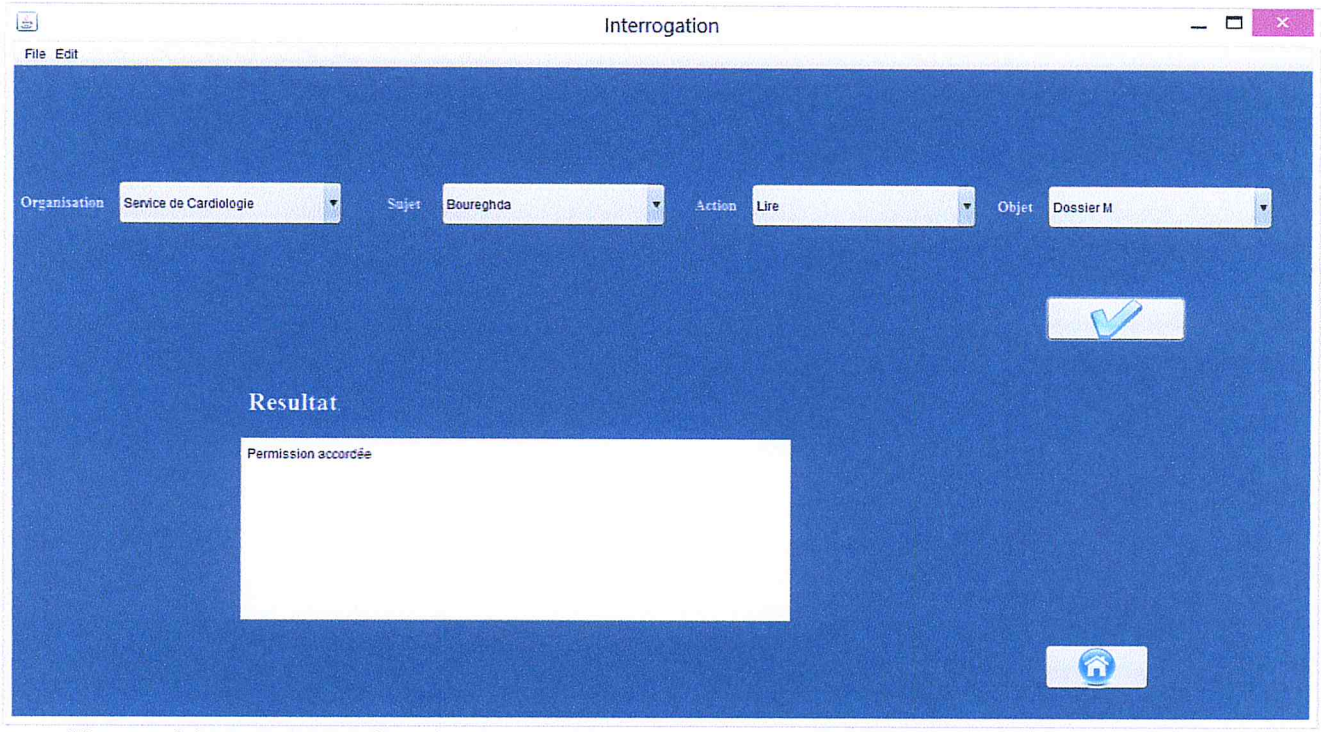

Figure 6.35 : Interrogation du système après Délégation de permission de licence dans le contexte défaut.

4.2. Délégation de Permission de Licence dans le contexte exception :

On fait une exception sur une permission de licence délégation pour l'annuler dans un contexte exceptionnel dans notre cas le contexte « vacances interrompues », on sélectionne la permission que laquelle on veut faire une exception. (**Figure 6.36**)

Avant cette exception le cessionnaire doit s'authentifier pour lui donner la main de faire une exception que sur les permissions qu'il a délégué et aussi pour vérifier que ce sujet est bien cessionnaire.

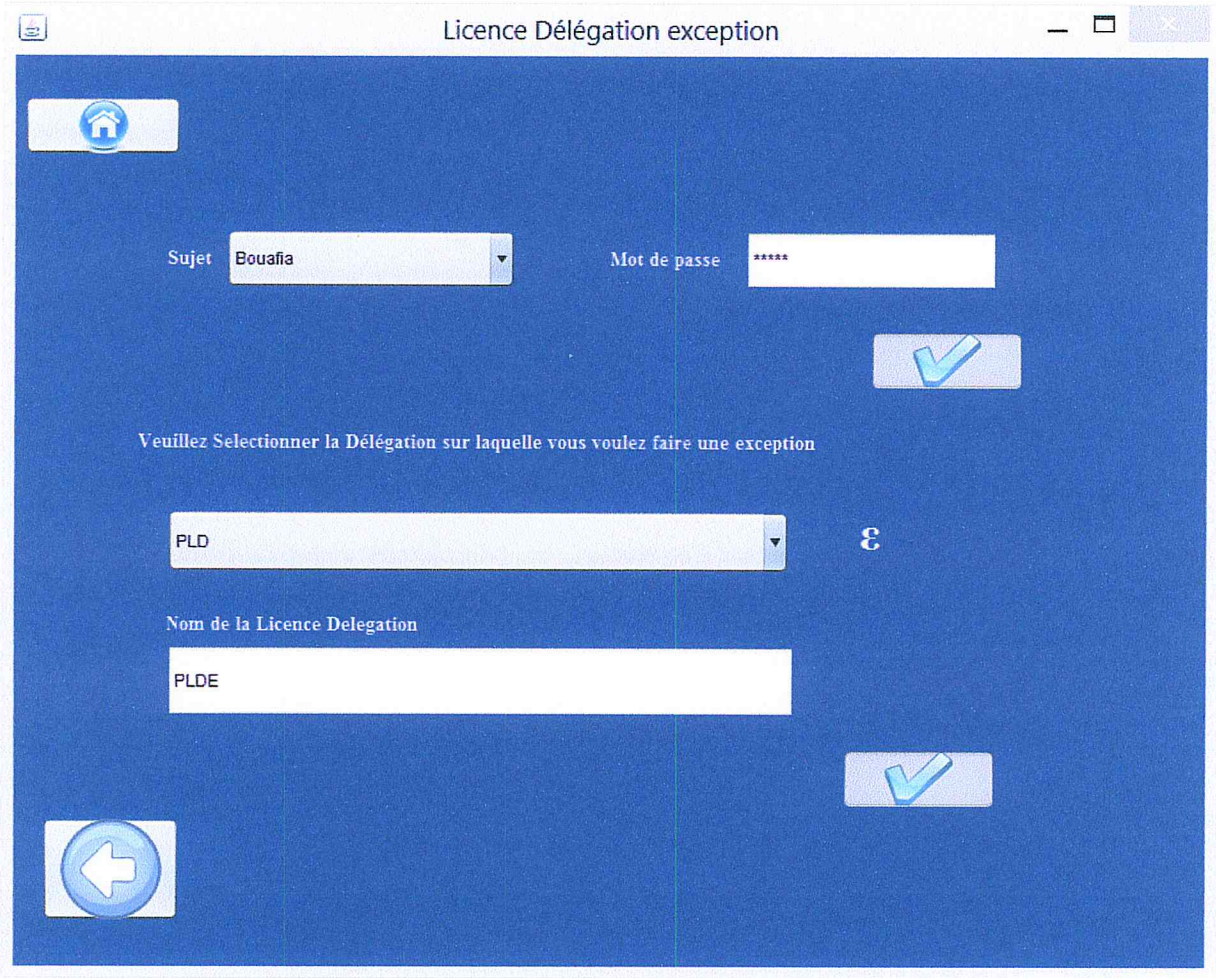

Figure 6.36 : Délégation de permission de licence dans le contexte exception.

Pour montrer que l'exception sur la délégation de permission de licence délégation est réussie on procède à l'interrogation du système. (**Figure 6.37**)

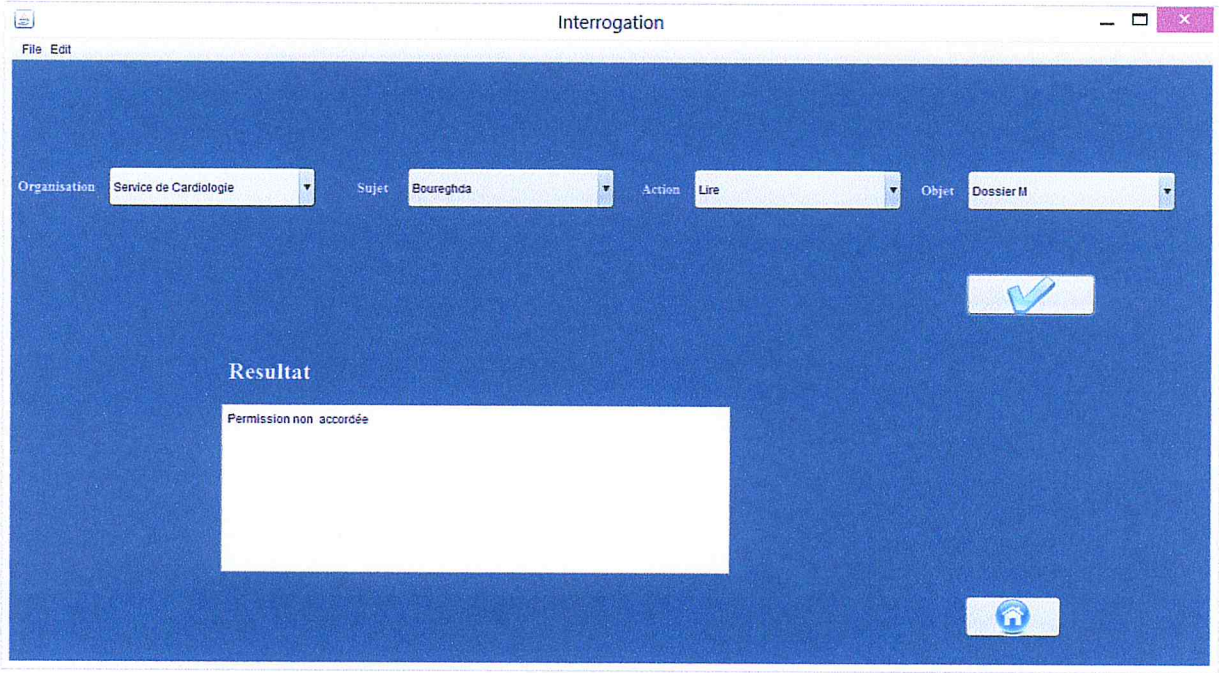

Figure 6.35 : Interrogation du système après Délégation de permission de licence dans le contexte exception.

4.5.14.5. Délégation de Permission de Licence-Transfer :

4.1. Délégation de Permission de Licence transfer dans le contexte défaut :
On procède à la délégation de permission de Licence- transfer qui est une délégation non monotone c'est-à-dire que le délégant perd le droit qu'il délègue dans un contexte défaut. dans notre cas ce contexte est « Congé maladie », le sujet « Bouafia » délègue une de ses permissions qui est la permission de lire l'objet dossier M au sujet « Boureghda » suivant l'instance permission de licence-transfer **PLT .(Figure 6.37)**
Avant cette délégation le cessionnaire doit s'authentifier pour lui donner la main de déléguer que les permissions qu'il obtient et pas d'autres, et aussi pour vérifier que ce sujet est bien cessionnaire. **(Figure 6.36)**

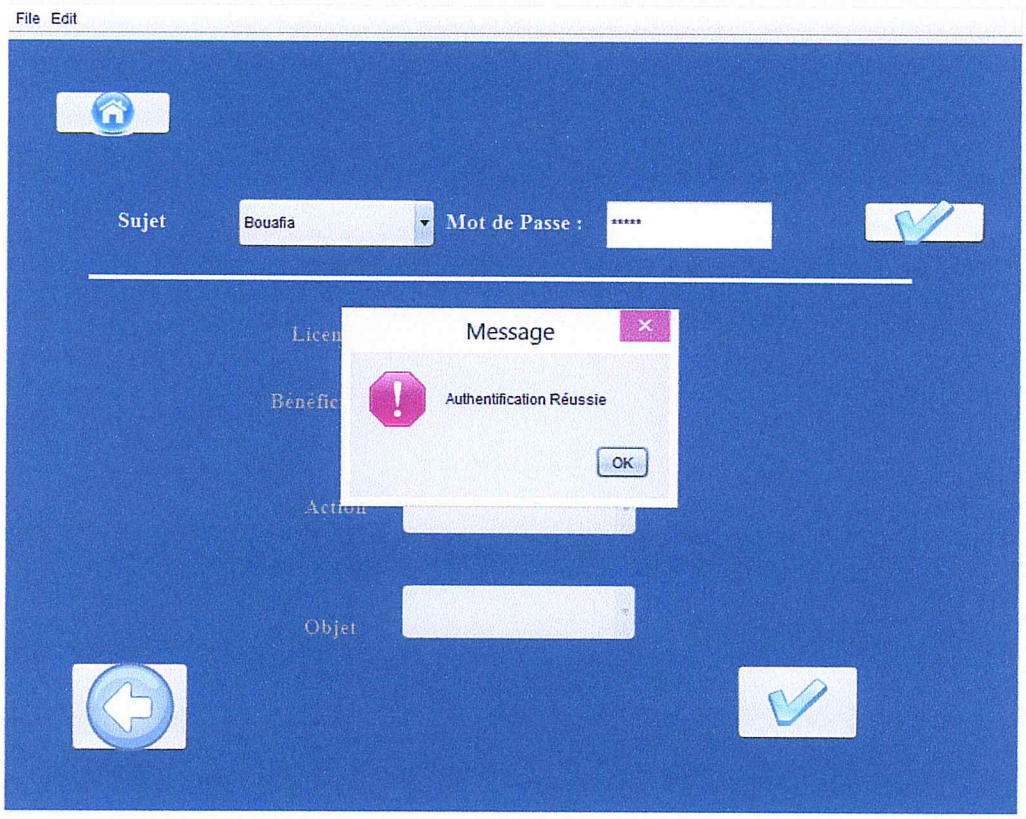

Figure 6.36 : Authentification cessionnaire Licence-transfer .

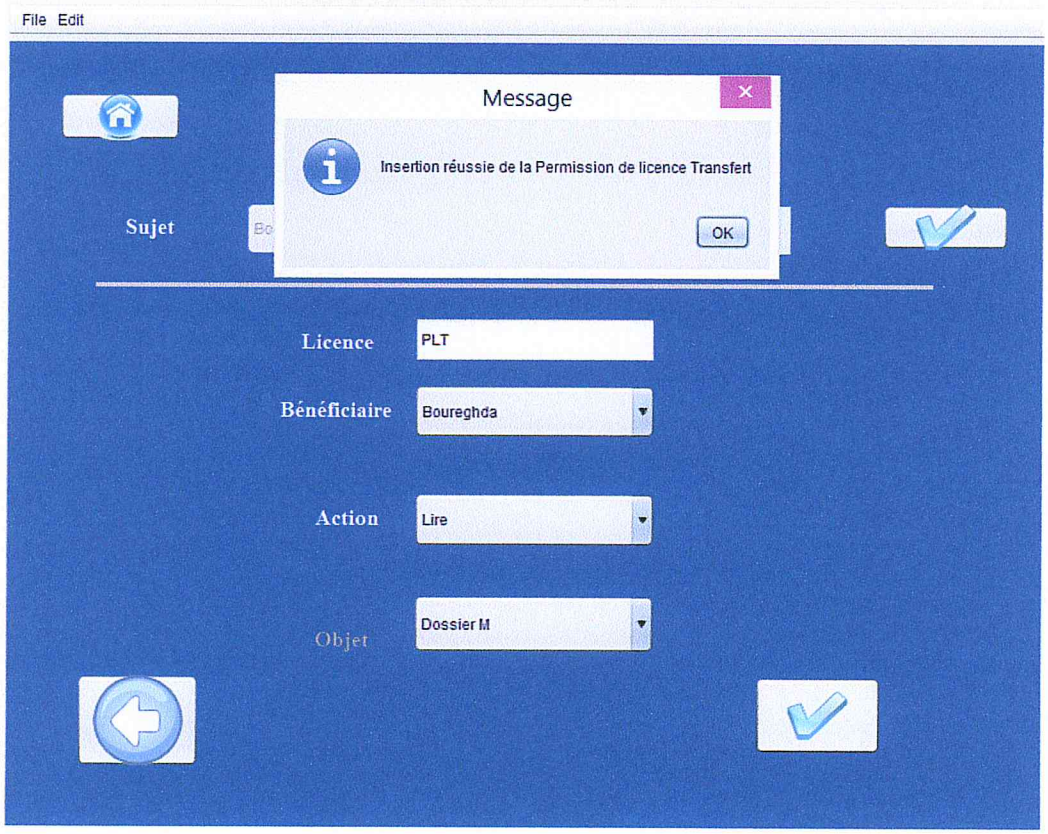

Figure 6.37 : Délégation de permission de licence-transfer dans le contexte défaut

Pour montrer que la délégation de permission de licence -transfer est réussie on procède à l'interrogation du système. Dans un premier temps pour montrer que le bénéficiaire a reçu le droit (**Figure 6.38**), et dans un deuxième pour montrer que le délégant a perdu ce droit **(Figure 6.39) .**

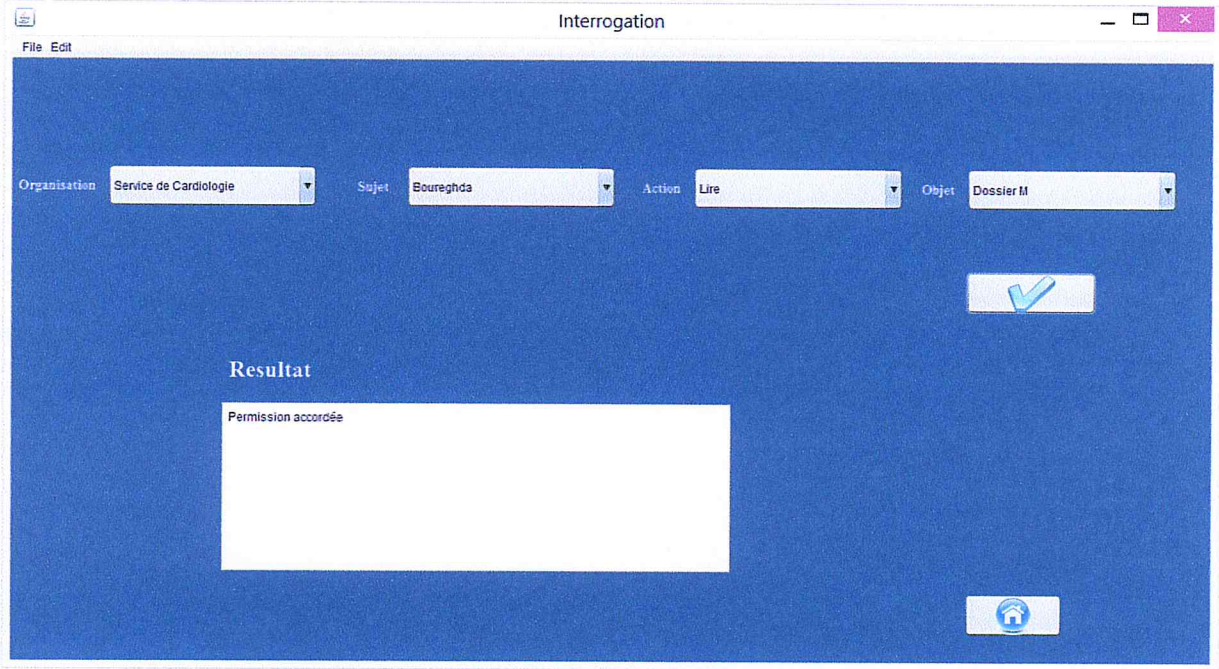

Figure 6.38 : Licence transfert dans le contexte défaut coté Bénéficiaire.

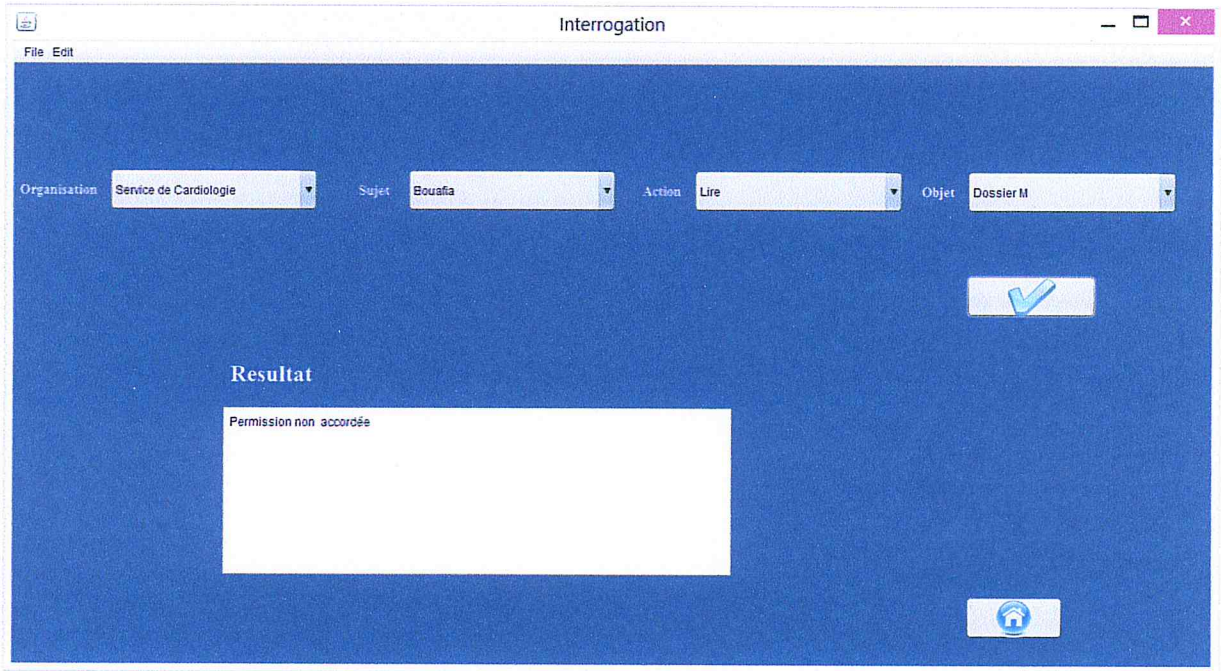

Figure 6.39 : Licence transfert dans le contexte défaut coté Délégant.

5.2. Délégation de Permission de Licence-transfer dans le contexte exception :
Quand on réalise une exception sur une permission de licence-transfer la délégation devient de type monotone c'est-à-dire que le cessionnaire récupère le droit qu'il avait transféré tout en laissant ce droit à la personne à qui il a délégué , lors de l'exception « retour de congé »le sujet « Bouafia » récupère une de ses permissions qui est la permission de lire l'objet dossier M tout en laissant cette permission au sujet « Boureghda » suivant l'instance **PLT_E (Figure 6.41).**
Avant cette exception le cessionnaire doit s'authentifier pour lui donner la main de récupérer que les permissions qu'il avait et pas d'autres, et aussi pour vérifier que ce sujet est bien cessionnaire. **(Figure 6.40)**

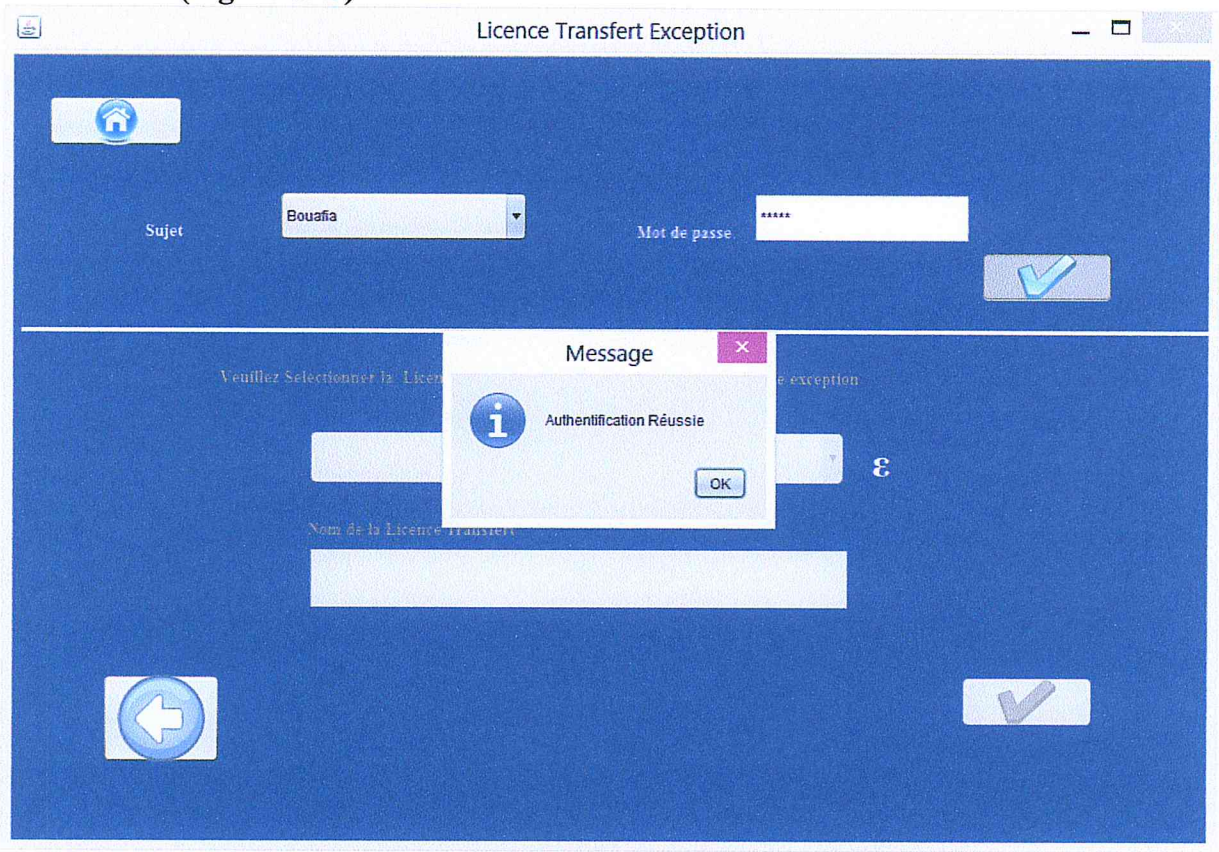

Figure 6.40 : authentification du cessionnaire Licence-transfer exception.

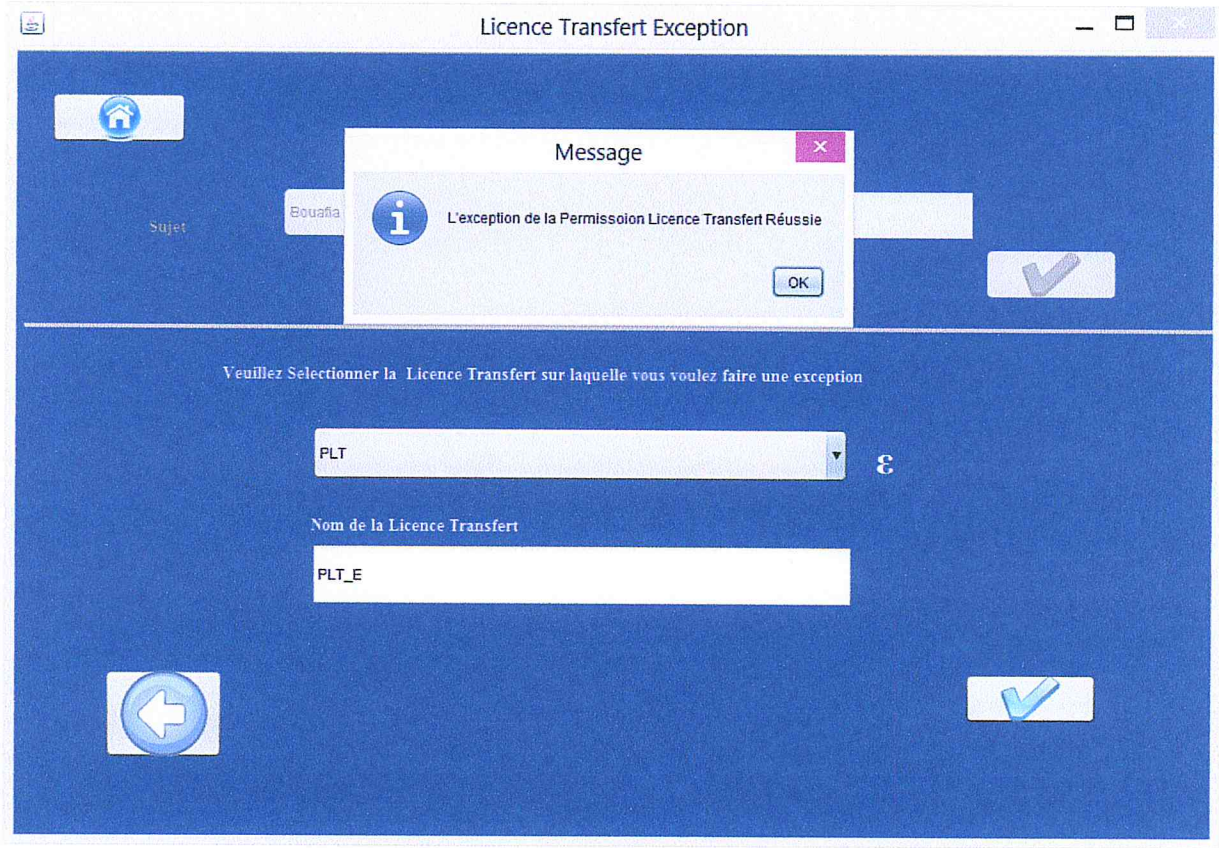

Figure 6.41 : Délégation de permission de licence-transfer dans le contexte exception.

Pour montrer que l'exception sur la délégation de permission de licence -transfer est réussie on procède à l'interrogation du système. Dans un premier temps pour montrer que le bénéficiaire a gardé le droit **(Figure 6.38)** on référence la première figure parce- qu'on a le même résultat, et dans un deuxième pour montrer que le délégant a récupéré ce droit **(Figure 6.42) .**

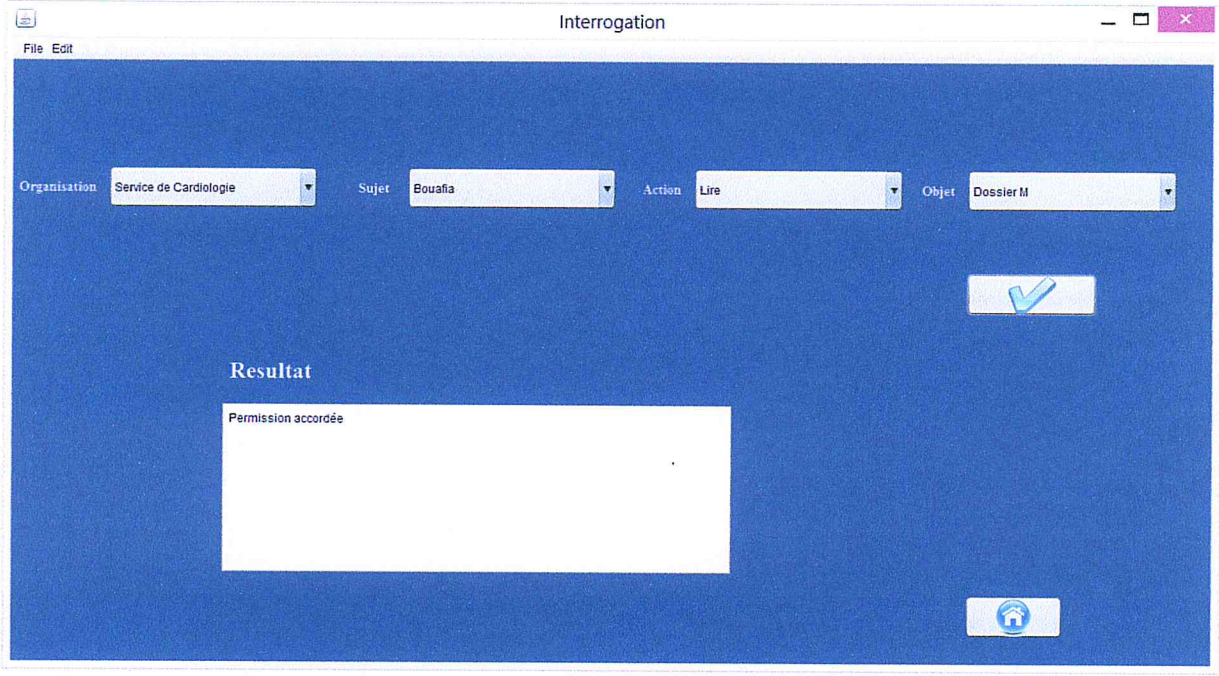

Figure 6.42 : Interrogation après Licence transfert dans le contexte exception .

4.5.14.6. Délégation de Permission Grant-option-Licence:

6.1. Délégation de Permission Grant-option-Licence dans le contexte défaut :
On procède à la délégation de permission Grant-option licence qui est une délégation n-pas c'est-à-dire que le délégant délègue son droit à un bénéficiaire, il lui délègue également le droit de pouvoir déléguer ses droits à un autre bénéficiaire et cette opération est valable jusqu'à ce que le niveau de délégation devienne nulle. Dans notre cas ce contexte est « Fonctionnement normal » et le n=3 donc on aura une délégation à 3-pas.
Le sujet « Bouafia » délègue une de ses permissions qui est la permission de créer l'objet dossier A au sujet « Boureghda » suivant l'instance permission de grant-option licence PGOL avec un niveau n=3, le sujet « Boureghda » recevra cette permission et deviendra à son tour cessionnaire avec un niveau n=2.(**Figure 6.43)**

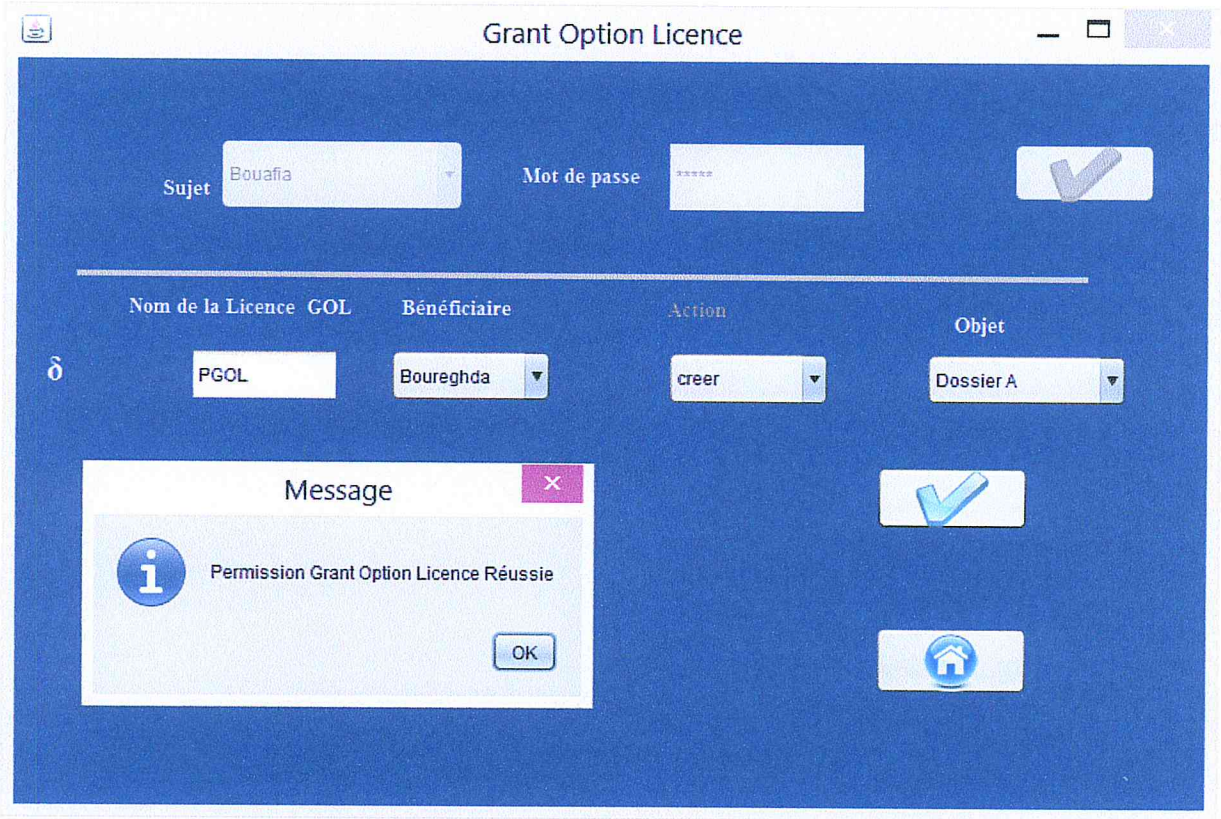

Figure 6.43 :Délégation Permission grant-option dans un contexte défaut au premier niveau

Le sujet « Bougherda » délègue la permission que le sujet « Bouafia » lui a délégué qui est la permission de créer l'objet dossier A au sujet « Ziraoui » suivant l'instance permission de grant-option licence **PGOL1** avec un niveau n=2, le sujet « Ziraoui» recevra cette permission et deviendra à son tour cessionnaire avec un niveau n=1.(**Figure 6.44)**

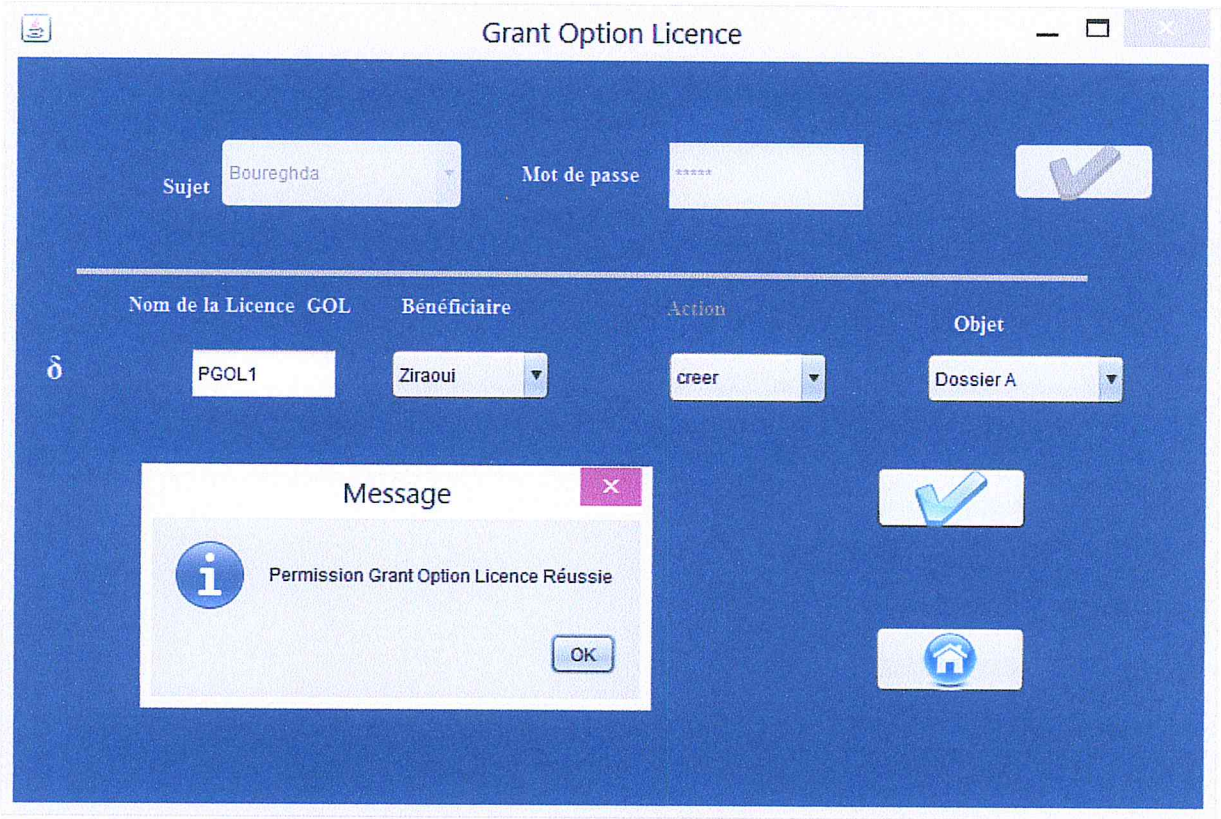

Figure 6.44 : Délégation Permission Grant-option dans un contexte défaut au deuxième niveau .

Le sujet «Ziraoui» délègue la permission que le sujet « Bougherda» lui a délégué qui est la permission de créer l'objet dossier A au sujet « Chettibi» suivant l'instance permission de grant-option licence **PGOL2** avec un niveau n=1, le sujet « Ziraoui» recevra cette permission et deviendra à son tour cessionnaire avec un niveau n=0.**(Figure 6.45)**

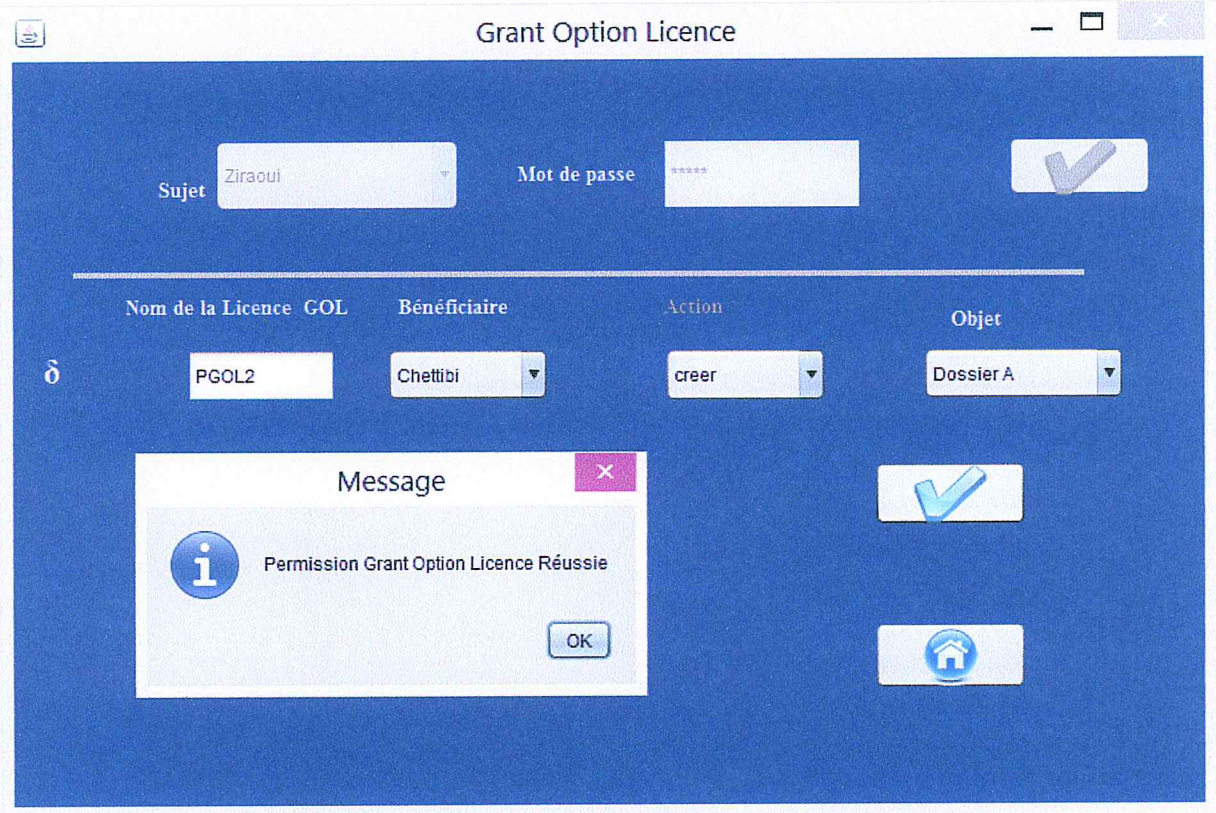

Figure 6.45 : Délégation Permission Grant-option dans un contexte défaut au troisième niveau.

Le sujet «Chettibi» tente de déléguer la permission que le sujet «Ziraoui» lui a délégué qui est la permission de créer l'objet dossier A au sujet « Benalama» suivant l'instance permission de grant-option licence **PGOL4** mais puisque son niveau est n=0 il n'a plus droit de déléguer.(**Figure 6.46)**

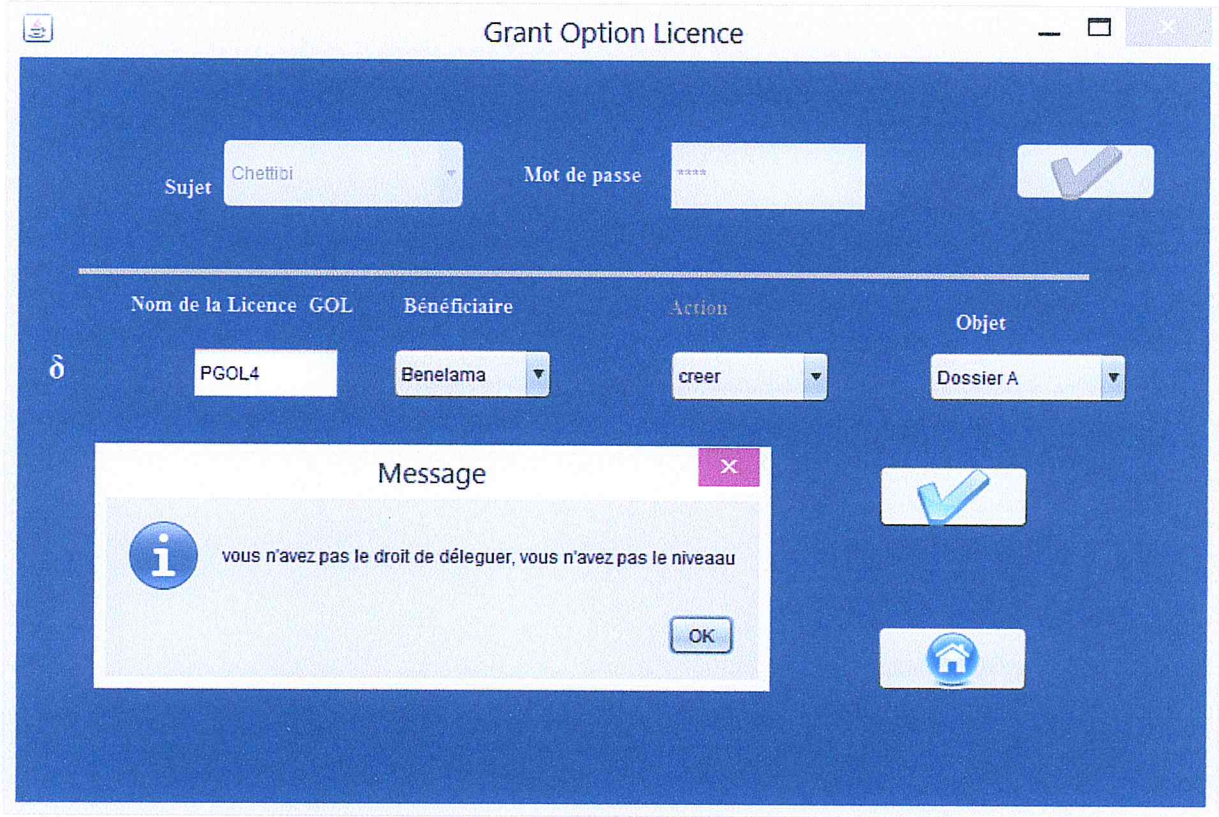

Figure 6.46 : Délégation Permission Grant-option dans un contexte défaut au dernier niveau.

Pour montrer que les délégations de permission précédentes ont bien été réussies on procède à l'interrogation du système. Cette interrogation se faisant comme dans les sections précédentes nous n'avons pas jugé nécessaire de l'illustrer par des figures.

6.2. Délégation de Permission Grant-option-Licence dans le contexte exception :

On procède à une exception sur une délégation de permission Grant-option licence qui est une délégation à n-pas , cette exception rendra la délégation à n-pas une délégation à 1-pas c'est-à-dire que le délégant délègue son droit à un bénéficiaire, et le droit de pourvoir déléguer à son tour sera supprimé . Donc on prend le niveau du délégant et on supprime les droits de tous les délégants dont le niveau est inférieur.

Dans notre cas l'exception est « état de crise » (**Figure 6.47**) .Si on prend l'exemple du délégant « Bouafia » dont le niveau est 3, on doit supprimer le droit de déléguer de chaque bénéficiaire devenu délégant dont le niveau est inférieur à 3 (**Figure 6.48**), et les droits de chaque bénéficiaires dont le niveau est inférieur à 2 .

Dans notre cas seul la délégation du sujet « Bouafia » au sujet « Boureghda »restera valable. (**Figure 6.49**)

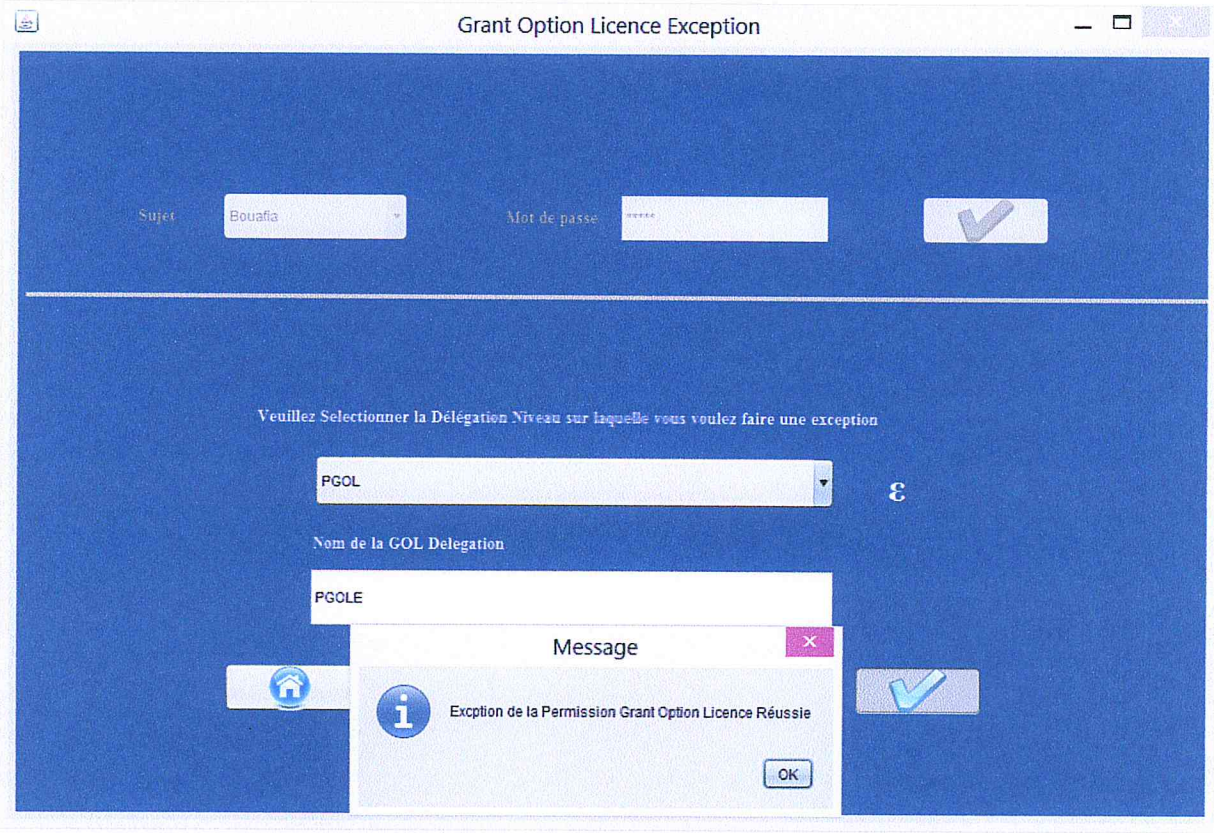

Figure 6.47: Délégation Permission Grant-option dans un contexte exception .

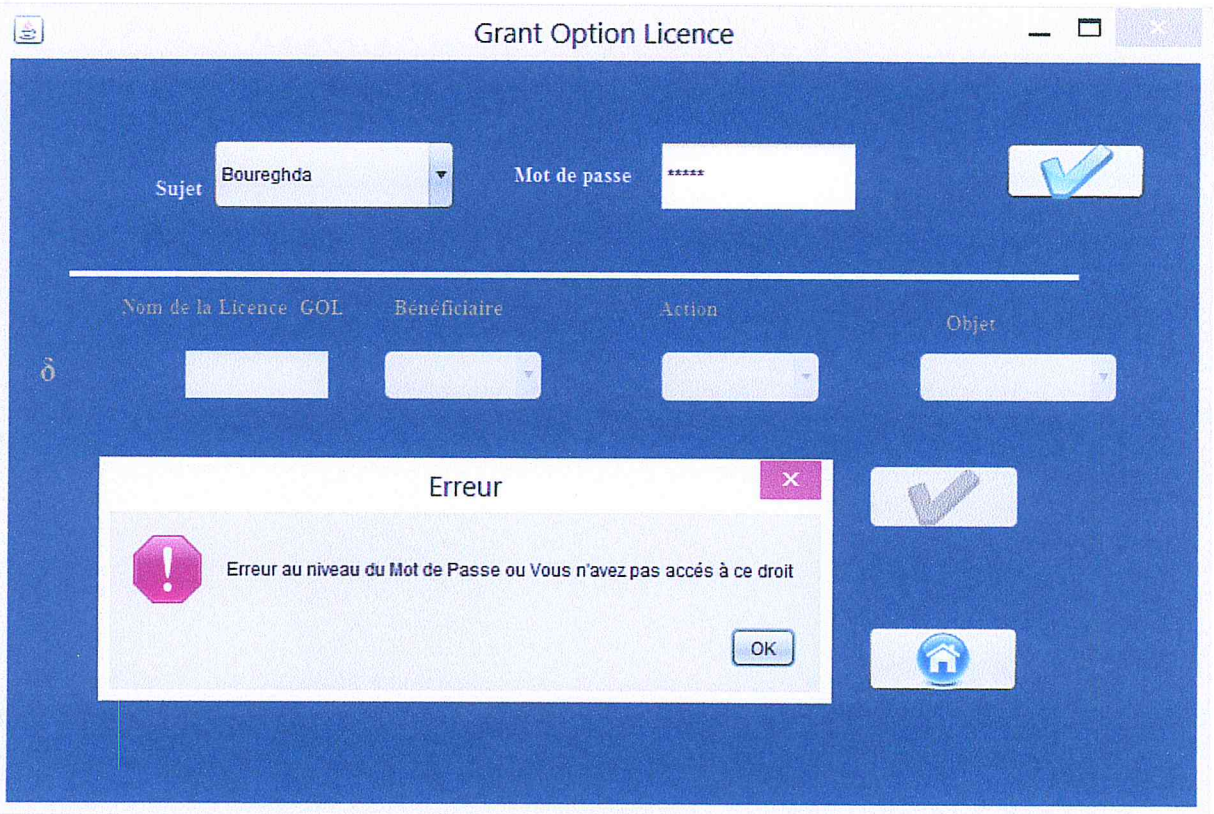

Figure 6.48: authentification dans une Délégation Permission Grant-option dans un contexte exception.

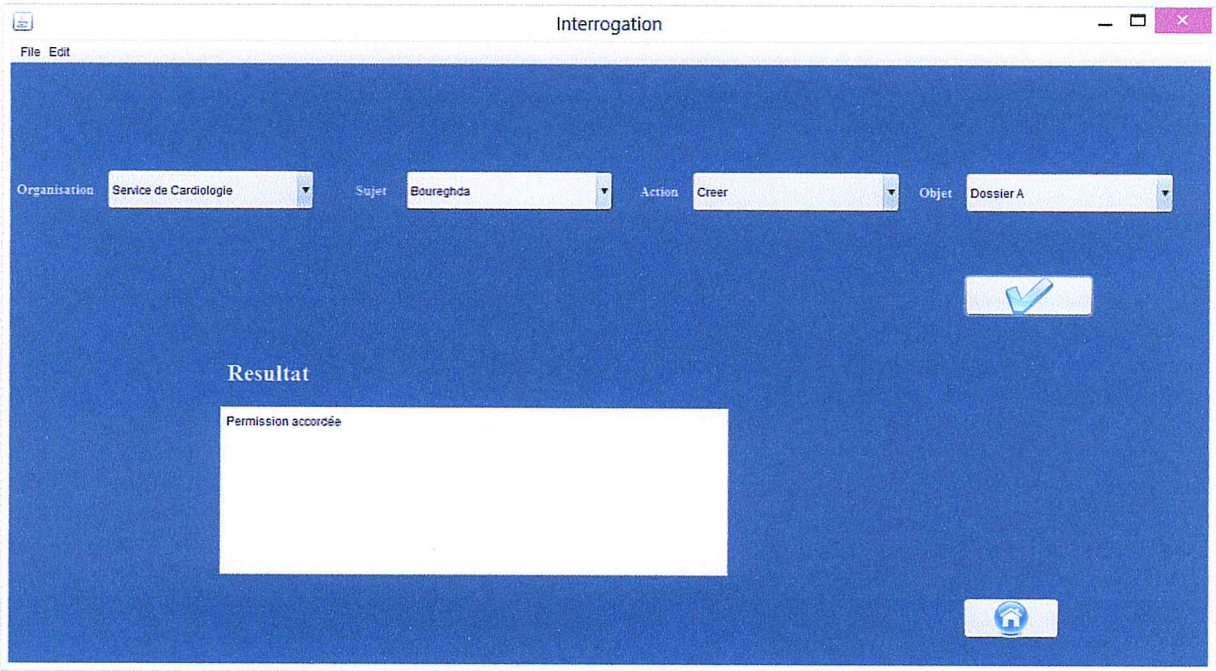

Figure 6.49 : Interrogation du système après une exception sur Délégation Permission Grant-option.

4.5.14.7. Révocation en cascade:

On procède à une révocation en cascade sur la délégation de permission Grant-option licence qui est une délégation à 3-pas (**Figure 6.50**), cette révocation supprimera toutes les délégations qu'elles soient directes ou indirectes c'est-à-dire la délégation de « Bouafia »à « Boureghda »,de« Boureghda»à « Ziraoui», et enfin « Ziraoui»à «Chettibi».
Pour montrer que la révocation a bien eu lieu on procède à l'interrogation du système .on illustre par la figure de l'interrogation du système dans le cas de la délégation entre le sujet « Ziraoui » et « Chettibi » **(Figure 6.51)** en précisant que les autres interrogations donnent le résultat escompté.

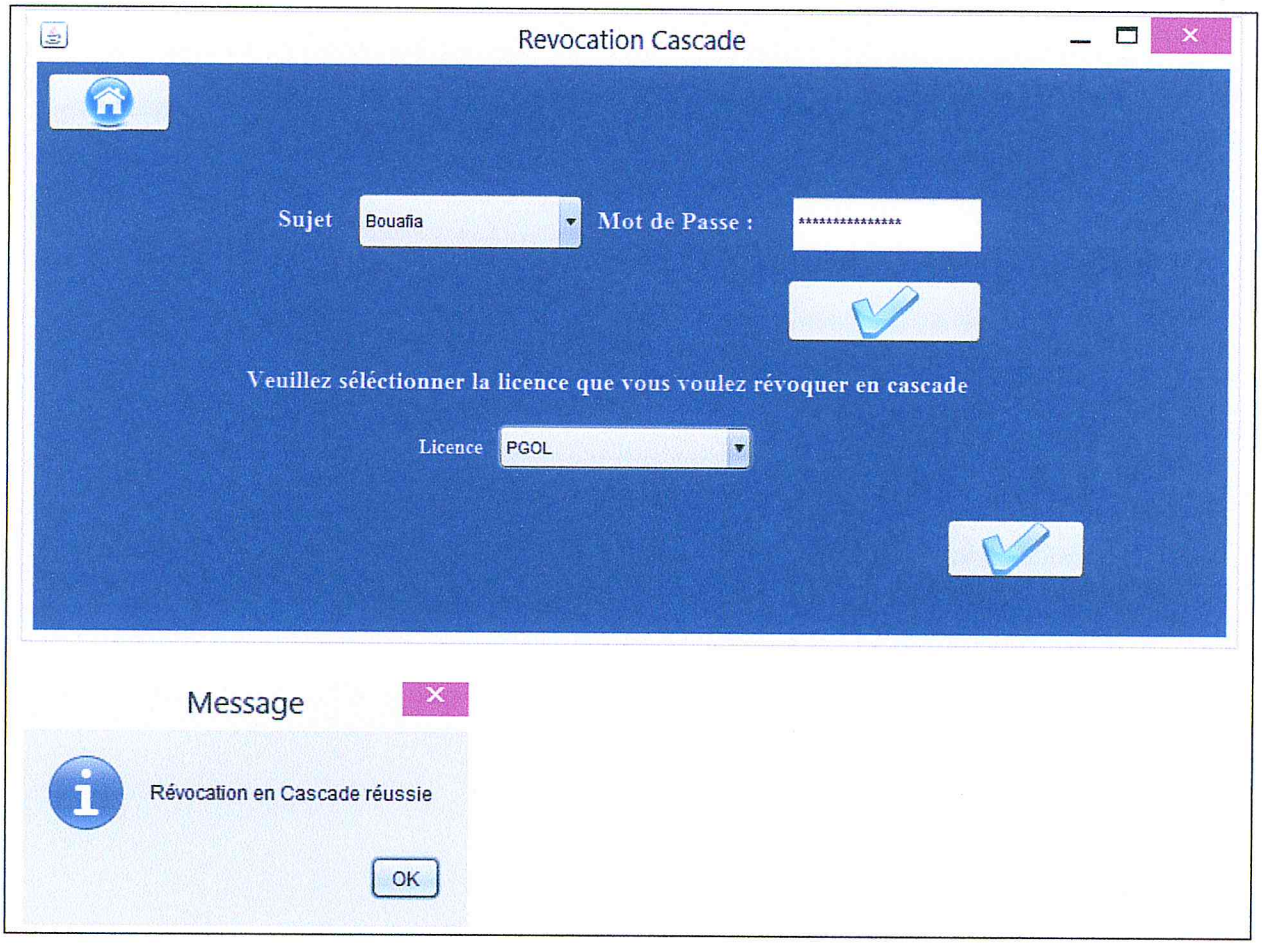

Figure 6.50 : Révocation en cascade de Permission Grant-option-Licence .

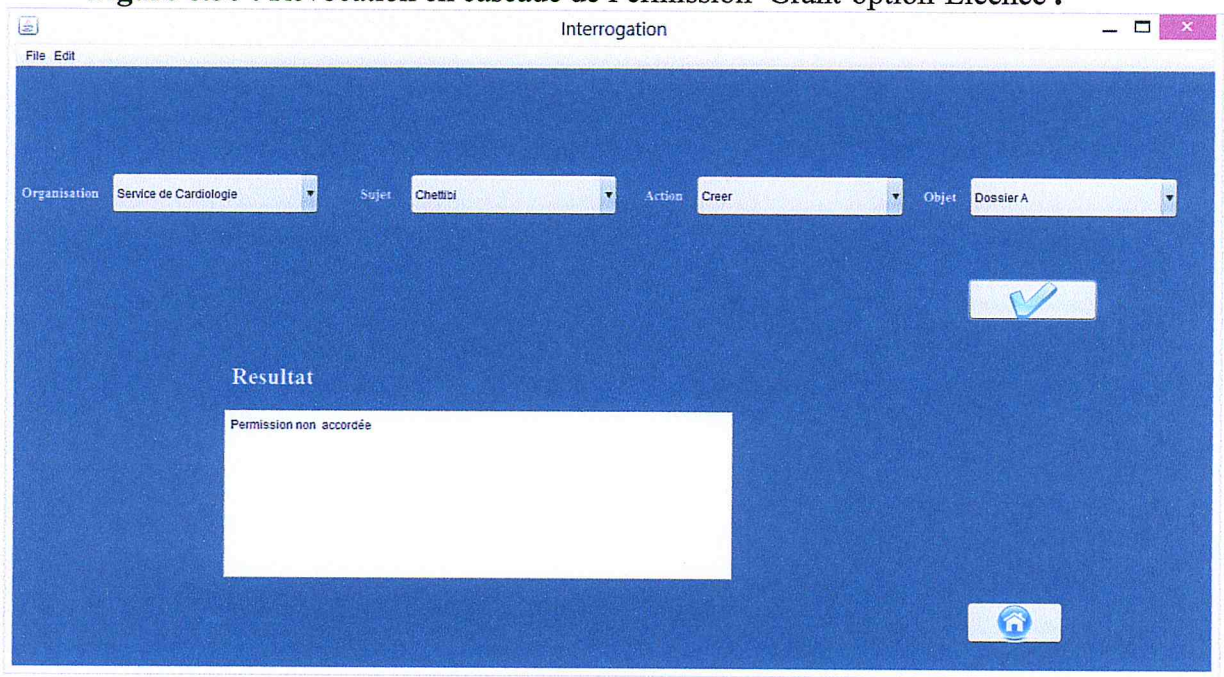

Figure 6.51 : Interrogation après une Révocation en cascade de Permission grant-option-Licence.

5. Conclusion

Dans ce chapitre nous avons commencé par présenter les outils servant au développement de l'application simulant notre modèle ensuite nous avons appliqué le modèle de contrôle d'accès DELEGATION-OrBACδε sur un système d'information médical existant et nous l'avons présenté à travers quelques interfaces graphiques pour en montrer les fonctionnalités. DELEGATION-OrBACδε regroupe deux parties : la première consiste en la création et la manipulation des différentes entités qui composent le système d'information médical. Nous avons détaillé les différentes opérations d'ajout, modification et suppression ainsi que les relations d'habilitation (entre rôle et sujet) considérant que les utilisations et considérations se font suivant le même principe .La deuxième partie se focalise sur l'introduction de permissions que ça soit les permissions abstraites ou alors les permissions de délégations et l'inférence déclenchée par celle-ci . Elle contient aussi un service d'interrogation qui facilite à l'utilisateur d'avoir une réponse rapide sur une requête de demande.

Conclusion et perspectives

Conclusion et Perspectives

Le travail présenté dans ce manuscrit concerne la réalisation d'un modèle de contrôle d'accès dynamique et contextuel formalisé avec la logique de description défaut et exception .On a nommé notre modèle **DELEGATION-OrBACδε.** Ce modèle s'est inspiré d'un modèle existant, OrBAC .Le but de notre travail fut principalement de modéliser la notion de délégation .Cette notion est particulièrement intéressante puisqu'elle permet de donner à un utilisateur particulier un privilège, sans donner ce privilège à toutes les personnes ayant le même rôle que lui. La délégation a été modélisée en utilisant les différentes entités et relations d'OrBAC , le fait de les utiliser nous a permis de spécifier la délégation sans avoir besoin d'ajouter de nouveaux composants au modèle OrBAC ni de modifier les existants.

Les trois principales phases qui ont constitué la modélisation et réalisation de notre modèle sont :

- ✓ La modélisation et création de la base de connaissances du modèle et les inférences sur cette base.
- ✓ La manipulation des différentes entités créées dans la phase précédente.
- ✓ L'expression de la délégation.

Pour formaliser notre modèle nous avons utilisé une Logique de description non monotone, qui intègre les opérateurs défaut (δ) et exception (ε), les logiques de description classiques n'étant pas suffisantes.

Cette étude nous a permis de mieux comprendre l'utilité de la logique de description augmentée des opérateurs défaut et exception et son application dans le cadre pratique tel qu'un modèle de contrôle d'accès ainsi que l'intérêt d'inclure la modélisation de la délégation malgré la complexité de celle-ci .

En perspective nous proposons d'étendre ce travail en considérant les points suivants :

- ❖ Prendre en compte d'autres contextes tels que le contexte temporel et spatial.
- ❖ Intégrer au modèle **DELEGATION-OrBACδε**, en plus des permissions les obligations et les recommandations.
- ❖ Etendre la prise en charge des types de délégations à tous les types existants puisque dans notre modèle nous avons géré que les principaux types.

Références bibliographiques

Bibliographie et Webographie

[1] :Anas Abou El Kalam, Rania El Baida, Philippe Balbiani, Salem Benferhat,Frédéric Cuppens, Yves Deswarte, Alexandre Miège,Claire Saurel, Gilles Trouessin, « ORBAC : un modèle de contrôle d'accès basé sur les organisations », projet RNRT MP6 (Modèles et Politiques de Sécurité des Systèmes d' Informations et de Communication en Santé et en Social).

[2] : F.Cuppens, N. Cupens-Boulahia & A.Miège, "Héritage de privilèges dans le modèle OrBAC: application dans un environnement réseau" Journées SSTIC, juin2004.

[3] :www.orbac.org.

[4] : Celine Coma ,"Administration d'une politique de contrôle d'accès" ,Janvier 2005.

[5] : Mokhtari Mohamed Amine ,Ghersi Cherifa « Mémoire pour l'obtention d'un Master « Délégation dans le contrôle d'accès avec logique de description » , juillet 2012.

[6] : Frederic cuppens , Alexandre Miège « Or-BAC ,Organisation Based Access control ",2004

[7]: Meriam Ben-Ghorbel-Talbi , Frédéric Cuppens ,Nora Cuppens-Boulahia , Adel Bouhoula "A delegation model for extended RBAC", pp.214,2010.

[8]: M. Minsky, "A framework for representing knowledge". Dans J. Haugeland (éditeur),Mind Design. The MIT Press, pp. 95-128, 1981.

[9]: W.A. Woods, "Understanding subsomption and taxonomy: a framework for progress", In Morgan-Kaufmann, editor, Principals of Semantic Networks, pp. 45- 94. J. Sowa, 1991.

[10]: F. Baader, D.L. McGuiness, D. Nardi and P.F. Schneider, "The Description logic handbook: Theory, Implementation and Applications", Cambridge university press (ISBN-13:9780521781763), 2008.

[11]: D. Tsarkov and I. Horrocks, "DL reasoner vs. first order prover". In Proc. Of the 2003 Description Logic Workshop (DL 2003) Volume. pp.152-159

[12]: L. Padgham, B. Nebel, "Combining classification and no monotonic inheritance reasoning: a first step". In t77th International Symposium on Methodologies for Intelligent Systems, pp. 15- 18, Norway. 1993.

[13] : N. Boustia et A. Mokhtari, "A dynamic access control model". In Applied Intelligence Journal, DOI 10.1007 /s 10489-010-0254-z, 2010.

[14] :Nardi, D. et Brachman, R. J., 2003. An introduction to description logics. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), The Description Logic Handbook : Theory, Implementation and Applications. Cambridge University Press, pp. 544.

[15] :Tsarkov, D. et Horrocks, I., 2003. DL reasoner vs. rst-order prover. Dans Proc. of the 2003 Description Logic Workshop (DL 2003) volume. pp. 152-159.

[16] :Schmidt-Schaub, M. et Smolka, G., 1991. Attributive concept descriptions with complements. Articial Intelligence 48 (1), 126

[17] :Baader, F. et Nutt, W., 2003. Basic description logics. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), The Description Logic Handbook : Theory, Implementation and Applications. Cambridge University Press, pp. 47100.

[18] :W.A. Woods, "Understanding subsomption and taxonomy: a framework for progress", In Morgan-Kaufmann, editor, Principals of Semantic Networks, pp. 45- 94. J. Sowa, 1991.

[19] :J. Quant, V. Royer, "Preference semantics for defaults in terminological logics", In Principals of knowledge Representation and Reasoning: 3rd International Conference, pp. 294- 305. Bernhard Nebel, Charles Rich and William Swartout, Cambridge 1992.

[20] : P. Coupey, C. Fouqueré, "Extending conceptual definitions with default knowledge", Comput Intell 13(2), 1997.

[21]: www.01net.com.

[22] :www.impactpc2b.com,Mai 2013.

[23] : N. Boustia, "Un formalisme de sécurisation des bases de données multi-niveaux", thèse de doctorat, Université des Sciences et de la Technologie Houari Boumediene (USTHB), Alger, 2011.

[24] : L. Benkezzim et M. Delleci, "*Etude et réalisation du raisonneur C-Classicδε*". Mémoire d'ingénieur, Université des Sciences et de la Technologie Houari Boumediene (US THB), Alger, 2009.

[25] :L.Padgham & T.Zhang, 1993 L. Padgham, T. Zhang. A terminological logic with defaults:a definition and an application. In 13the international Joint Conference on Artificial Intelligence, pp 663-668, Chambery, France. 1993 .

[26] : [Brachman, 1983] R.J. Brachman. Krypton: Integrating terminology and assertion. In Third National Conference on Artificial Intelligence, page 31-35, Washington, 1983.

[27] : W.W. Cohen and H. Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In International Conference of Knowledge Representation and Reasoning, pp 121-133. 1994.

[28] : Franz Baader, Deborah L. McGuinness, Daniele Nardi, Peter F. Patel-Schneider : THE DESCRIPTION LOGIC HANDBOOK : Theory, implementation, and applications . Cambridge University Press, Cambridge, 2007 pp 47.

[29] : Fournier-Viger, Philippe (2005) "Un modèle de représentation des connaissances à trois niveaux de sémantique pour les systèmes tutoriels intelligents". Mémoire de maîtrise (M.Sc.), Université de Sherbrooke, Sherbrooke, Canada.

[30] : V.Ventos. C-CLASSICδε: une logique de description pour la définition et l'apprentissage de concepts avec défaults et exception. Thèse, université Paris-Nord, France, 1997.

Annexe

Algorithme de calcul de subsomption SOLLIN2

SOLLIN2 est un algorithme de calcul de subsomption, de type normalisation comparaison. Il est composé d'une étape de normalisation des descriptions, suivi d'une étape de comparaison purement syntaxique de formes normales obtenues de la première étape. Cet algorithme défini par *VENTOS* est complet, correct et polynomial. Les preuves sont données dans **[30]**.

Étant donné deux descriptions C et D de C-CLASSIC$_{\delta\varepsilon}$. Répondre à la question « D subsume t-il C ? », en utilisant l'algorithme de SOLLIN2 consiste en première partie à calculer les formes normales de C et C Π D, en appliquant la procédure de Normalisation et de les comparer en appliquant la procédure Comparer.

Algorithme SOLLIN2 (SubsumeTest)

En entrée : C et D deux descriptions de concepts étendues[1] de C-CLASSIC$_{\delta\ \varepsilon}$.

En sortie : "Oui" ou "Non" à la question "D subsume t-il C ? "

```
Début
    {* Calcul des formes normales*}
        Fn (C) ←——— Normalisation (C)
        Fn (C Π D) ←——— Normalisation (C Π D)
        {*Traitement du concept plus spécifique (BOTTOM)*}
  Si Fn (C) = bo2 alors réponse ←——— Oui
  sinon
    Si Fn (C Π D) = bo alors réponse ←——— Non
        Sinon {*Comparaison des formes normales*}
            Comparer (Fn (C)σ, Fn (C Π D)σ ,rep1)
                si rep1 = "Oui" alors  Début
                                       Comparer (fn (C)δ, fn (C Π D)δ, rep2)
                                       Réponse ←——— rep2
                                       Fin
                sinon Réponse ←——— Non
                Fin Si
    FinSi
  FinSi
Fin
```

Procédure de calcul de la forme normale : Normalisation

L'algorithme qui suit est un algorithme de calcul de la forme normale (procédure de Normalisation), qui distingue les termes de la forme $\forall$ r : C (avec C $\neq$ T), et ceux comportant le connecteur Π (appelés termes composés), et les autres termes appelés termes simples. L'algorithme utilise la procédure calcul-index et la procédure union-fn (notée $\otimes$).

Procédure de Normalisation

En entrée : D une description étendue d'un concept de J-CLASSICδε .
En sortie : Fn (D), la forme normale de D.

Début
{*Cas de base: termes simples*}
Si D est T **alors** Fn (D) ← < (UNIV, MIN-R, MAX-R, ϕ, ϕ, ϕ) , (UNIV, MIN-R, MAX-R, ϕ, ϕ, ϕ) >
Si D est ⊥ **alors** Fn (D) ← bo[1]
Si D est un concept primitif P
alors Fn (D) ← < (UNIV, MIN-R, MAX-R, {P}, ϕ, ϕ), (UNIV, MIN-R, MAX-R, {P}, ϕ, ϕ), >
Si D est ONE-OF E
alors Fn (D) ← < (E, MIN-R, MAX-R, ϕ, ϕ, ϕ), (E, MIN-R, MAX-R, ϕ, ϕ, ϕ), >
Si D est MIN u
alors Fn (D) ← < (UNIV, u, MAX-R, ϕ, ϕ, ϕ), (UNIV, u, MAX-R, ϕ, ϕ, ϕ), >
Si D est MAX u
alors Fn (D) ← < (UNIV, MIN-R, u, ϕ, ϕ, ϕ), (UNIV, MIN-R,u, ϕ, ϕ, ϕ), >
Si D est r FILLS E (E ≠ ϕ)
alors Fn (D) ← < (UNIV, MIN-R, MAX-R, ϕ, {<r, E, |E|, NOLIMIT,t>}, ϕ),
(UNIV, MIN-R, MAX-R, ϕ, {<r, E, |E|, NOLIMIT,t>}, ϕ)>
Si D est r FILLS ϕ **alors** Fn (D) ← t[2]
Si D est r AT-LEAST n (n >0)
alors Fn (D) ← < (UNIV, MIN-R, MAX-R, ϕ,{<r, ϕ, n,NOLIMIT, t >} ,ϕ),
(UNIV, MIN-R, MAX-R, ϕ,{<r, ϕ, n,NOLIMIT, t >} ,ϕ)>
Si D est r AT-LEAST 0 **alors** Fn (D) ← t
Si D est r AT-MOST n (n >0)
alors Fn (D) ← < (UNIV, MIN-R, MAX-R, ϕ,{<r, ϕ,0, n, t >} ,ϕ),
(UNIV, MIN-R, MAX-R, ϕ,{<r, ϕ, 0,n, t >} ,ϕ)>
Si D est r AT-MOST 0 **alors** Fn (D) ← < (UNIV, MIN-R, MAX-R, ϕ,{<r, ϕ,0, 0, bo >} ,ϕ),
(UNIV, MIN-R, MAX-R, ϕ,{<r, ϕ,0, 0, bo >} ,ϕ)>
Si D est ∀R : T **alors** Fn (D) ← t
Si D est ∀R : ⊥ **alors** Fn (D) ← <(UNIV, MIN-R, MAX-R, ϕ,{<r, ϕ,0, 0, bo >} ,ϕ),
(UNIV, MIN-R, MAX-R, ϕ,{<r, ϕ,0, 0, bo >} ,ϕ)>

[1] bo: est une constante qui représente la forme normale du concept le plus spécifique, BOTTOM (⊥).
[2] t: représente la forme normale du concept le plus général, le concept top. (le premier cas dans l'algorithme)

```
{*Termes composés*}
Si D est ∀ r : C

Alors  Début Fn (C) ← Normalisation (C)
       Si Fn (C) = t alors Fn (D) = t
       Sinon  Si Fn (C) = b0
                    alors fn (D) ← <(UNIV, MIN-R, MAX-R, ϕ,{<r, ϕ,0, 0, bo >} ,ϕ),
                                    (UNIV, MIN-R, MAX-R, ϕ,{<r, ϕ,0, 0, bo >} ,ϕ)>
              Sinon Fn (D) ← <(UNIV, MIN-R, MAX-R, ϕ,{<r, ϕ,0, |Cσ.dom|, C>},ϕ),
                                    (UNIV, MIN-R, MAX-R, ϕ, {<r, ϕ,0, |Cσ.dom|, C >} ,ϕ)>
              FinSi
       FinSi

       Fin

Si D est δC alors   Début
                    Fn (C) ← Normalisation (C)
                    Fn (D) ← < (UNIV, MIN-R, MAX-R, ϕ, ϕ, ϕ), Fn (C )δ >
                    Fin
Si D est C ε alors  Début
                    Fn (D)σdom ← UNIV
                    Fn (D)σmin ← MIN-R
                    Fn (D)σmax ← MAX-R
                    Fn (D)σπ ← ϕ
                    Fn (D)σr ← ϕ
                    Fn (D)σε ← calcul-index (Fn (C) δ)
                    Fn (D)δdom ← Fn (C)δdom
                    Fn (D)δmin ← Fn (C)δmin
                    Fn (D) δmax ← Fn (C)δmax
                    Fn (D)δ π ← Fn (C)δ π
                    Fn (D)δr ← Fn (C)δr
                    Fn (D)δε ← Fn (D)σε U Fn ( C)δε
                    Fin
Si D est A Π B alors Début
                    Fn (A) ← Normalisation (A)
                    Fn (B) ← Normalisation (B)
                    Fn (D) ← Fn (A) ⊗ Fn (B) {*union de deux formes normales*}
                    Fin
FIN
```

Procédure calcul-index :

Le fait qu'une exception doit toujours portée sur un défaut, cela veut dire qu'à chaque fois que l'utilisateur saisie une exception dans sa description, il faut vérifier que cette exception est déjà portée sur un défaut.

Annexe

Pour un gain d'espace mémoire, nous définissons la procédure calcul-index qui consiste à coder les sextuplets défaut de la description, et cela en construisant une table d'index comportant des couples (nombre, sextuplet), où nombre correspond au codage du sextuplet défaut qui sera utilisé dans le calcul de la forme normale.
Cette façon de coder est importante dans le cas où la description comprend une série d'ε imbriqués.

Procédure calcul-index (S) (S est un sextuplet)

Début
Si il existe un couple (n, s) dans la table d'index **alors** calcul-index(s) ← n
Sinon on crée un nouveau couple dans la table d'index (m, s)
où m=nombre de couples dans la table d'index +1
calcul-index ← m
FIN

Procédure d'union de deux formes normales ⊗ :

Elle permet de faire l'union de deux sextuplets. Cette union se fait sur chacun de leurs champs (dom, min, max,π r,ε).

Procédure union-fn

En entrée : deux sextuplets A et B:
(a.dom, a.min, a.max, a.π, a.r, a.ε) et (b.dom, b.min, b.max, b.π, b.r, b.ε)
En sortie : (u.dom, u.min, u.max, u.π, u.r, u.ε)le sextuplet résultant de l'union de A et B

Début
u.dom ← a.dom ∏ b.dom
u.min ← maxi(a.min, b.min)
u.max ← mini (a.min, b.min)
u.π ← a.π U b.π
u.ε ← a.ε U b.ε
u.r ← ϕ
Pour tout <r, fills1, least1, most1, C1> ∈a.r **Faire**
Si ∃< r, fills2, least2, most2, C2 >∈b.r
u.r ← u.r U < r, fills, least, most, C> avec:
fills ← fills1 U fills2
C ← C1 ⊗ C2
least ← maxi (least1, least2, |fills|)
most ← mini (most1, most2, |Cσ.dom|)
Si least = |Cσ.dom| **alors** fills ← Cσ.dom
Sinon **Si** most =|fills| **alors** Cσ.dom ← fills
Cδ.dom ← Cδ.dom ∏ Cσ.dom
FinSi
FinSi
u.r ← u.r U < r, fills1, least1, most1, C1>

```
    FinSi
    Pour tout < r, fills, least, most, C> ∈b.r telque ∄ d'élément de nom r dans a.r
            u.r ← u.r U < r, fills, least, most, C>
    FinPour
FinPour
FIN
```

Procédure de comparaison de deux formes normales (Comparer) :

La procédure Comparer représente la seconde partie de l'algorithme de $Sub_{\delta\varepsilon}$. Cette procédure permet de tester l'égalité entre deux sextuplets t1 et t2.
t1 (respectivement t2) est de la forme: (ti.dom, ti.min, ti.max, ti.π, ti.r, ti.ε) ,avec i = 1 (respectivement 2).

Procédure Comparer (T1, T2, Réponse) (equals)

```
Début
  Si (T1. dom ≠[3] T2.dom) ou (T1.min ≠T2.min) ou (T1.max ≠T2.max) ou (T1.π≠[4]T2.π)
      ou  (T1.ε ≠ T2.ε)
  Alors Réponse ← Non

  Sinon {*Comparaison des champs rôles*}
    Début
      Appel Comparer-rôle (T1.r, T2.r, rep)
      Réponse ← rep
    Fin
  Fin Si
FIN
```

3 Il s'agit d'une égalité ensembliste (l'ordre des éléments n'intervient pas).
4 Les champs ε sont des ensembles de nombres qu'il suffit de comparer. Il est inutile d'étendre les nombres par Des sextuplées leur correspondant dans la table index

3) Supprimer P (respectivement ¬P) dans Cδπ si ¬P (respectivement P) est dans Cσπ.

En entrée : c, la description du concept C en CClassic$_{\delta\ \varepsilon}$.
En sortie : Result, la forme d'héritage du concept C.

Result ← ⟨ ($c_{\sigma\ \pi}$, Ø, Ø), ($c_{\delta\ \pi}$, Ø, Ø) ⟩
/ Remplacer chaque exception de niveau pair par un défaut et l'insérer dans $c_{\sigma\pi}$ */*
Pour tout y ∈ $c_{\sigma\ \pi} \cup c_{\delta\ \pi}$
    faire
        Result ← Result ∪ *remplace* (y, $c_{\sigma\ \pi}$)
    fin Pour
/ Appeler récursivement la procédure schèma_héritage avec restriction de valeur de rôle*/*
Pour tout ⟨r, p⟩ ∈ $c_{\sigma\ \pi}$
    faire
    Result ← Result ∪ ⟨ (Ø, ⟨ r, *schéma_héritage*(p) ⟩, Ø), (Ø, Ø, Ø) ⟩
    Fin Pour

Pour tout ⟨ r, p⟩ ∈ $c_{\delta\ \pi}$
    faire
    Result ← Result ∪ ⟨ (Ø, Ø, Ø), (Ø, ⟨ r, *schéma_héritage*(p) ⟩, Ø) ⟩
    fin Pour

Soit Result est) (*Result*$_{\sigma\ \pi}$, *Result*$_{\sigma}$r, *Result*$_{\sigma\ \varepsilon}$), (*Result*$_{\delta\ \pi}$, *Result*$_{\delta}$r, *Result*$_{\delta\ \varepsilon}$) ⟩

Pour tout x ∈ *Result*$_{\sigma\ \pi}$
    faire
    supprimer ¬ x de *Result*$_{\delta\pi}$
    fin Pour

Pour tout ¬ x ∈ *Result*$_{\sigma\ \pi}$
    faire
    supprimer x de *Result*$_{\delta\pi}$
    fin Pour

Return Result

Corrections

Chapitre	Section	Ligne	Numéro de Page	Erreur	Correction
01	4.1. Hiérarchie de rôles	1	12	4.1. Hiérarchie de rôles	4.1. Hiérarchie de rôles **:**
	4.2. Hiérarchie d'activités	1	12	4.2. Hiérarchie d'activités	4.2.Hiérarchie d'activités **:**
	4.3. Hiérarchie de vues :	1	12	hiérarchies dépendant	hiérarchies **dépendantes**
02	2.2.La délégation dans AdOrBac :	5	16	per missions	permissions
	3. Définition générale de la délégation:	12	16	de de	**de**
	4.2. Délégation monotone/non-monotone :	5	18	si	**Si** (Majuscule)

Corrections

	4.2. Délégation monotone/non-monotone :	7	18	perds	perd
	4.4. Délégation totale/partielle :	4	19	ces droits	**S**es droits
	4.5. Délégation par agent/auto-active :	8	19	n'a besoin	n'a **pas** besoin
	4.5. Délégation par agent/auto-active :	10	19	il est obligé de passé	Il est obligé de pass**er**
	4.5. Délégation par agent/auto-active	11	19	celui	Celui **ci**
	4.9. Révocation de la délégation simple/en cascade :	17	20	a 2 pas	**à** 2 pas
	4.9. Révocation de	18	20	les droits	Les droits **donnés**

Corrections

	la délégation simple/en cascade :			données	
	5.Conclusion :	7	21	nous sommes basées nous	nous sommes **basés** nous
03	3.1 .Niveau terminologique (TBox) :	1	23	du la	**De** la
	3.3.Les composants base des Logiques description :	6	25	qui restreints	Qui **sont** restreints
	3.5.2.3 .Complétion :	1	28	la complétion	**L**a complétion
	3.5.2.3 .Complétion :	1	28	propriétés hérités	propriétés **héritées**
	6.2. Description des connecteurs δ et ε :	15	32	*P. Coupey et C. Fouqueré*	*P. Coupey et C. Fouqueré* **ont introduit**

Corrections

	6.2. Description des connecteurs δ et ε :	15	32	présente	**Re**présente
	6.4 .Caractéristiques d'ALδε : • Points forts :	1	34	des défaut	des défauts
	6.4 .Caractéristiques d'ALδε : • Points forts :	2	34	on tant	**En** tant
	6.4 .Caractéristiques d'ALδε : • Points forts :	5	34	classificatin	classification
	7.Conclusion :	7	34	la puissance d'expressivité	La **faible** puissance d'expressivité
04					

Corrections

	3.Syntaxe de la logique CClassicδε :	16 et 19	36	à l'échelle de Richter	**Sur** l'échelle de Richter
05	1. Introduction :	5	41	comprends	comprend
	2. Représentation de connaissances :	2	41	définit	défini
	3.1.3. Axiome de définition d'activités :	13	45	la connaissance	**L**a connaissance
	3.1.4.2. Axiome de définition de hiérarchie d'activités :	4	46	ou	où
	3.1.7.1.2. La vue rôle délégation :	16	52	définit	définie
06	4.1. Interface principale :	5	65	deux	d'eux

Corrections

	4.4. Suppression d'une instance de concept :	5	70	qui rentre	qui rentrent
	4.2. Délégation de Permission de Licence dans le contexte exception :	3	92	que laquelle	Sur laquelle
	6.2. Délégation de Permission Grant-option-Licence dans le contexte exception :	9	103	chaque bénéficiaires	chaque bénéficiaire
	5. Conclusion	8	107	considérant que les utilisations et considérations se font suivant le même principe	**Ainsi que** les utilisations et considérations **qui** se font suivant le même principe