ORBAC : un modèle de contrôle d'accès basé sur les organisations

Anas Abou El Kalam[1] Rania El Baida[2] Philippe Balbiani[2]
Salem Benferhat[3] Frédéric Cuppens[2] Yves Deswarte[1]
Alexandre Miège[4] Claire Saurel[5] Gilles Trouessin[6]

CRIL[3] ENST[4] Ernst & Young Audit[6] IRIT[2] LAAS-CNRS[1] ONERA[5]

Résumé : 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 dans le domaine médical. Dans cet article, nous proposons 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.

Abstract: None of the classical access control models such as DAC, MAC, RBAC, TBAC or TMAC is fully satisfactory to model security policies that are not restricted to static permissions but also include contextual rules related to permissions, prohibitions, obligations and recommendations. This is typically the case of security policies that apply to the health care domain. In this paper, we suggest a new model that provides solutions to specify such contextual security policies. This model, called Organization based access control, is presented using a formal language based on first-order logic.

[1] Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 4.
[2] Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse Cedex 4.
[3] CRIL – Université d'Artois, Rue Jean Souvraz, SP18, 62307 Lens.
[4] Ecole Nationale Supérieur des Télécommunications, 46 rue Barrault, 75634 Paris Cedex 13.
[5] Office National d'Etudes et de Recherches Aérospatiales - Centre de Toulouse BP 4025, 31055 Toulouse Cedex 4.
[6] Ernst & Young Audit, 1 place Alphonse Jourdain, 31000 Toulouse.

1 Introduction

Plusieurs modèles de contrôle d'accès ont été proposés : DAC [14], MAC [2, 6], RBAC [15, 12, 11], TBAC [17] ou TMAC [18]. Aucun de ces modèles n'est entièrement satisfaisant au regard des difficultés rencontrées pour mettre en œuvre, au sein d'une organisation, une politique de sécurité qui prendrait en compte les points suivants :

1. Des règles qui spécifient des permissions ou des interdictions contextuelles. En effet, il est fréquent d'avoir des règles de sécurité spécifiques à un certain contexte. Dans le domaine médical par exemple, les médecins ont des permissions spéciales dans des contextes spécifiques comme l'urgence (voir section 5).
2. Des règles qui spécifient des obligations ou des recommandations. Les modèles de contrôles d'accès classiques sont généralement restreints aux permissions. Certains incluent des interdictions, et plus récemment des modèles de politique de sécurité ont ajouté des obligations [9,5].
3. Des règles spécifiques à l'organisation. En particulier, l'organisation peut être structurée en plusieurs sous organisations qui ont chacune leur propre politique de sécurité. Le modèle devra ainsi proposer un moyen de spécifier au sein d'une même organisation plusieurs politiques de sécurité.

L'objet de cet article est de présenter un modèle qui tente de prendre en compte ces différents points. Le concept d'organisation est central dans ce nouveau modèle. L'organisation est un des paramètres des règles de sécurité de sorte qu'il soit possible de gérer à la fois plusieurs politiques de sécurité associées à différentes organisations. Le modèle n'est pas restreint aux permissions mais permet également de définir des interdictions, des obligations et des recommandations.

L'article est organisé comme suit : les sections 2 et 3 présentent des modèles de contrôle d'accès discrétionnaire et basé sur les rôles. La section 4 décrit notre modèle ORBAC. Dans la section 5, nous définissons un langage basé sur la logique du premier ordre utilisé pour définir une politique de sécurité ORBAC. La section 6 développe un exemple de politique de sécurité basé sur ce langage. Dans la section 7 nous évoquons comment exprimer différents types de contraintes. Enfin, la section 8 conclut l'article.

2 Le contrôle d'accès discrétionnaire

Harrison, Ruzzo et Ullman [13] définissent un modèle de politique de sécurité, le modèle HRU, qui s'applique sur des sujets, des objets et des actions. De manière intuitive, un sujet est une entité active alors qu'un objet est un conteneur d'information. Dans le contexte d'un système d'information, les sujets incluent les utilisateurs du système et généralement les processus exécutés pour le compte de ces utilisateurs. De plus, l'ensemble des objets inclut les entités actives, ainsi que les entités passives telles qu'un système d'information, des fichiers ou des répertoires. Les actions offrent aux sujets un accès direct aux objets. Elles correspondent généralement à des actions élémentaires comme "lire" ou "écrire". Dans ce modèle, la politique de sécurité est réduite à l'expression des permissions ; ces dernières étant des relations entre les sujets, les objets et les actions. Elles sont représentées dans la matrice *A* des permissions. Si *s* est un sujet et *o* est un objet alors, $A(s, o)$ définit l'ensemble des actions α que le sujet *s* est autorisé à faire sur l'objet *o*.

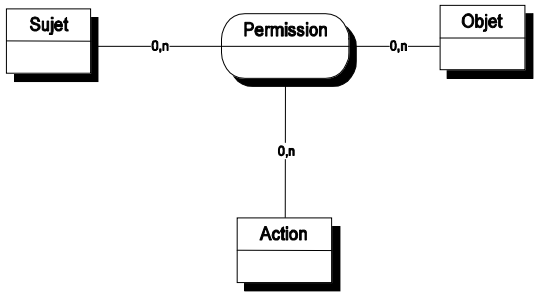

Figure 1 : le modèle HRU

Dans le modèle HRU, la politique de sécurité s'exprime à travers l'énumération dans la matrice des permissions de tous les triplets $<s, o, \alpha>$. Si de nouveaux objets, de nouveaux sujets ou de nouvelles actions sont ajoutés au système d'information, il est alors nécessaire d'enregistrer toutes les permissions accordées à ces nouvelles entités. Par conséquent, la mise à jour d'une politique de sécurité exprimée par ce modèle est quelque peu fastidieuse. Enfin, ce modèle ne permet pas d'exprimer des interdictions, des obligations ou des recommandations.

3 Le contrôle d'accès basé sur les rôles

Dans le modèle de contrôle d'accès basé sur les rôles, ou RBAC, la politique de sécurité ne s'applique pas directement à des utilisateurs comme dans le modèle précédent. Le rôle est ici le concept central de la politique de sécurité [12]. D'un côté des permissions sont accordées aux rôles ; de l'autre les utilisateurs se voient affecter un ou plusieurs rôles. Les utilisateurs obtiennent les permissions accordées aux rôles qu'ils jouent (figure 2).

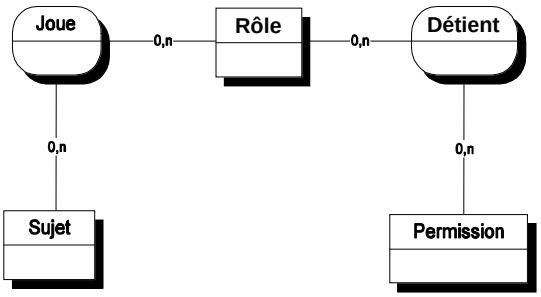

Figure 2 : Le modèle RBAC

Il est possible de raffiner ce modèle en incluant les concepts de session et de hiérarchie de rôles. Dans une même session, un utilisateur a la possibilité de ne pas activer tous ses rôles, mais uniquement le sous-ensemble de ses rôles nécessaires à la réalisation de la tâche à accomplir. La hiérarchie de rôles a cela d'utile qu'elle permet de mettre en place un mécanisme d'héritage des permissions entre les rôles et simplifie d'autant l'administration de ce modèle. Le modèle RBAC est complété par des contraintes. La séparation des pouvoirs, par exemple, peut être exprimée à l'aide d'une contrainte indiquant qu'un utilisateur n'est pas autorisé à jouer simultanément certains rôles dans une même session, comme ceux d'anesthésiste et de chirurgien lors d'une opération.

Les inconvénients du modèle RBAC sont les suivants. Tout d'abord, le concept de permission est primitif. En effet, dans le modèle RBAC, rien n'est dit sur l'usage ou la structure des permissions, considérant qu'ils sont dépendants de l'application concrète du modèle. Nous pensons à l'inverse qu'il serait préférable d'ajouter au modèle une structure générique de permission. Le concept de hiérarchie de rôles est quelque peu ambigu. Il est en général incorrect de considérer que la hiérarchie de rôles correspond à la hiérarchie organisationnelle. Par exemple, le directeur d'un hôpital a un rôle administratif supérieur au rôle de médecin. Pour autant, un directeur d'hôpital n'est pas nécessairement un médecin. Ainsi, il serait incorrect de considérer que le directeur de l'hôpital hérite des permissions du rôle de médecin, comme celle d'opérer par exemple. Enfin, la distinction entre le concept de rôle et celui de groupe est floue. Le groupe est un concept qui a été introduit avant la définition du modèle RBAC. Il y eut beaucoup de discussion à propos des différences entre le contrôle d'accès basé sur les groupes et RBAC. Le modèle ORBAC propose, dans la section 4, de clarifier ce point.

Ajoutons qu'il n'est pas possible dans le modèle RBAC d'exprimer des permissions qui dépendent du contexte. Plus précisément, si une certaine permission est accordée à un rôle, alors tous les utilisateurs qui jouent ce rôle héritent de cette permission. Par conséquent, il n'y a aucun moyen de spécifier qu'un médecin n'a la permission d'accéder au dossier médical d'un patient que si ce dernier est son patient [1, 7]. De plus, comme nous l'avons déjà évoqué dans le cas du modèle HRU, il est uniquement possible de définir des permissions. Enfin, l'application du modèle RBAC à la définition d'une politique de sécurité d'un système contenant plusieurs organisations fait apparaître d'autres limites de ce modèle.

4 Le contrôle d'accès basé sur l'organisation

Dans cette section, nous présentons notre modèle ORBAC en s'appuyant sur un langage diagramatique basé sur le modèle entité-relation. La section 5 présente le même modèle en utilisant un langage formel basé sur la logique du premier ordre. En conformité avec le modèle entité-relation, des attributs peuvent être associés aux entités et aux relations. Considérant que ces attributs sont généralement spécifiques au domaine d'application, nous n'en faisons pas état ici.

4.1 Les organisations

L'entité centrale dans notre modèle est l'*organisation*. Dans le domaine médical, nous pouvons considérer les organisations suivantes : "la clinique privée du Languedoc", "le service des urgences de l'hôpital Purpan", "l'unité des soins intensifs de l'hôpital Rangueil", etc. Une organisation peut être vue comme un groupe structuré 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 un certain accord entre les sujets pour former une organisation.

4.2 Les sujets et les rôles

L'entité *Sujet* est utilisée différemment selon les modèles de sécurité. Dans notre modèle ORBAC, un sujet peut être soit une entité active, c'est-à-dire un utilisateur, soit une organisation. Par exemple, "Jean", "Marie", "Pierre", etc., peuvent être des sujets, tout comme les organisations "département comptable de la clinique privée du Languedoc", "le service des urgences de l'hôpital Purpan", etc. Dans notre modèle, l'entité *Rôle* est utilisée pour structurer le lien entre les sujets et les organisations. 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, nous introduisons une relation entre ces entités : la relation *Habilite* (figure 3). Si *org* est une organisation, *s* est un sujet

et *r* est un rôle, alors *Habilite*(*org*, *s*, *r*) signifie que *org* habilite le sujet *s* à jouer le rôle *r*.

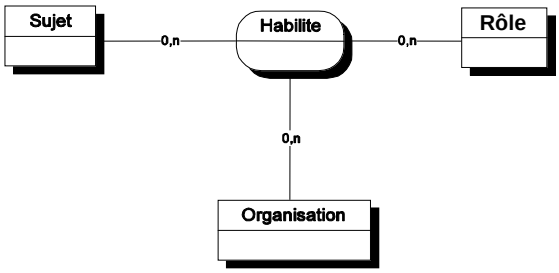

Figure 3 : La relation *Habilite*

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, notre modèle définit une relation ternaire entre les organisations, les sujets et les rôles. Les deux exemples suivants illustrent le fait que les sujets sont soit des utilisateurs, soit des organisations :

- *Habilite*(*Purpan*, *Jean*, *cardiologue*) : "l'hôpital Purpan habilite Jean dans le rôle cardiologue" et
- *Habilite*(*Rangueil*, *ICU31*, *unité_des_soins_intensifs*) : "l'hôpital Rangueil habilite l'unité ICU31 dans le rôle d'unité des soins intensifs".

4.3 Les objets et les vues

Dans notre modèle, l'entité *Objet* représente principalement les entités non actives comme les fichiers, les courriers électroniques, les formulaires imprimés, etc. 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 nous 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, nous considérons qu'une entité comparable au rôle pour les sujets est nécessaire pour les objets. Nous l'appelons : 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. 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, nous avons besoin d'une relation qui lie ces trois entités : la relation *Utilise* (figure 4). Si *org* est une organisation, *o* est un objet et *v* est une vue, alors *Utilise*(*org*, *o*, *v*) signifie que *org* utilise l'objet *o* dans la vue *v*.

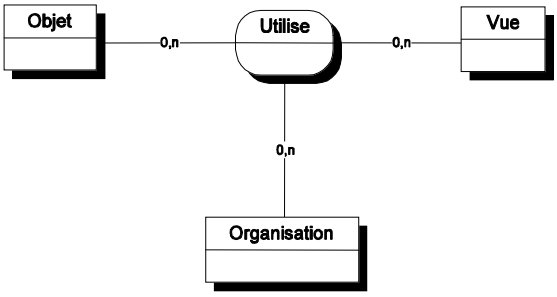

Figure 4 : La relation *Utilise*

Notre modèle 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. La vue "dossier médical" peut être définie à l'hôpital Purpan comme un ensemble de documents Word, et comme un ensemble de documents Latex à l'hôpital Rangueil :

- *Utilise*(*Purpan*, *F31.doc*, *dossier_médical*) : "L'hôpital Purpan utilise F31.doc comme un dossier médical" et
- *Utilise*(*Rangueil*, *F32.tex*, *dossier_médical*) : "L'hôpital Rangueil utilise F32.tex comme un dossier médical".

4.4 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 notre modèle, l'entité *Action* englobe principalement les actions informatiques comme "lire", "écrire", "envoyer", etc. De la même manière que dans les sections 4.2 et 4.3 où les rôles et les vues sont des abstractions des sujets et des objets, nous définissons une nouvelle entité 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 notre modèle, 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* (figure 5) sera utilisée pour associer les entités *Organisation*, *Action* et *Activité*. Plus précisément, si *org* est une organisation, α est une action et *a* est une activité, alors *Considère*(*org*, α, *a*) signifie que l'organisation *org* considère l'action α *comme* faisant partie de l'activité *a*.

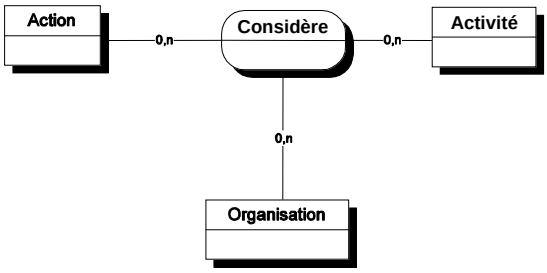

Figure 5 : La relation *Considère*

Remarquons que là encore nous définissons une relation ternaire. L'objectif est de pouvoir caractériser des organisations qui structurent différemment les mêmes activités. Si nous considérons l'activité "consultation". Cette activité peut correspondre, dans l'organisation hôpital Purpan, à l'action "lire" un fichier, mais peut tout aussi bien correspondre à l'action "select" sur une base de données dans l'hôpital Rangueil.

- *Considère*(*Purpan*, *lire*, *consultation*) : "l'hôpital Purpan considère *lire* comme une consultation" et
- *Considère*(*Rangueil*, *select*, *consultation*) : "l'hôpital Rangueil considère *select* comme une consultation".

4.5 Politique de sécurité (première définition)

En utilisant les entités et les relations introduites dans les sections précédentes, nous pouvons à présent définir des politiques de sécurité appliquées à telle ou telle organisation. Une politique de sécurité réglemente les accès au système à travers des permissions, des interdictions, des obligations et des recommandations. Nous ne traitons 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 et les activités. Plus précisément, si *org* est une organisation, *r* est un rôle, *a* est une activité et *v* est une vue, alors *Permission*(*org*, *r*, *a*, *v*) signifie que l'organisation *org* accorde au rôle *r* la permission de réaliser l'activité *a* sur la vue *v*. Prenons l'exemple de l'hôpital Purpan qui accorde au rôle "secrétaire médicale" la permission de réaliser l'activité "création" sur la vue "dossier administratif". Cette règle de sécurité est exprimée comme suit : *Permission*(*Purpan*, *secrétaire_médicale*, *création*, *dossier_administratif*). Considérons à présent la règle suivante : *Permission*(*Purpan*, *médecin*, *consultation*, *dossier_médical*) ; elle indique que l'hôpital Purpan permet aux médecins de consulter les dossiers médicaux. Pourtant, il est très probable que l'hôpital Purpan ne souhaite pas exprimer exactement cette règle, mais plutôt que dans des circonstances normales, un médecin à la permission de consulter les dossiers médicaux de ses propres patients uniquement. Le modèle ORBAC ne permet pas en l'état d'exprimer une telle règle. Nous proposons, pour résoudre ce problème, d'ajouter une nouvelle entité à notre modèle : l'entité *Contexte*.

4.6 Les contextes

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", "médecin traitant", 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 nouvelle relation appelée *Définit* (figure 6) telle que : si *org* est une organisation, *s* est un sujet, α est une action, *o* est un objet et *c* est un contexte, alors *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.

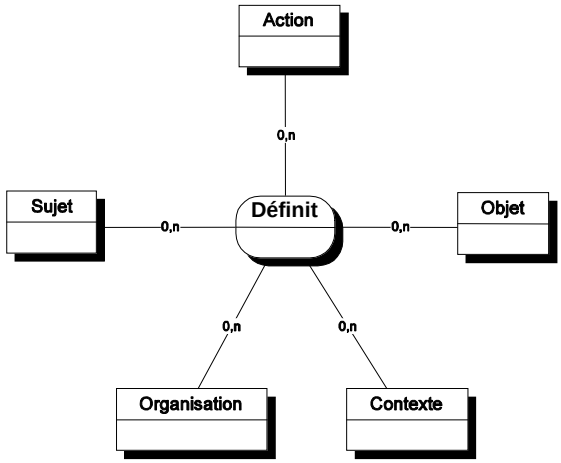

Figure 6 : La relation *Définit*

Les conditions requises pour qu'un contexte donné soit lié, dans une certaine organisation, aux sujets, aux objets et aux actions sera formellement spécifié par des règles logiques. Ceci sera abordé dans la section 5. Pour l'heure, considérons les faits suivants : *Définit*(*Purpan*, *Jean*, *lire*, *F31.doc*, *urgence*) et *Définit*(*Rangueil*, *Marie*, *lire*, *F32.tex*, *médecin_traitant*). Si le premier fait est vrai, alors Jean 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 Marie 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.

4.7 Politique de sécurité (définition finale)

Maintenant que nous avons présenté l’entité *Contexte*, nous pouvons à présent donner la définition finale d’une politique de sécurité dans le modèle ORBAC. La relation *Permission* correspond à une relation entre les organisations, les rôles, les vues, les activités et les contextes. Les relations *Interdiction*, *Obligation* et *Recommandation* sont définies de la même manière (figure 7). Si *org* est une organisation, *r* est un rôle, *a* est une activité, *v est* une vue et *c* est un contexte, alors *Permission*(*org*, *r*, *a*, *v*, *c*) signifie que l’organisation *org* accorde au rôle *r* la permission de réaliser l’action *a* sur la vue *v* dans le contexte *c*.

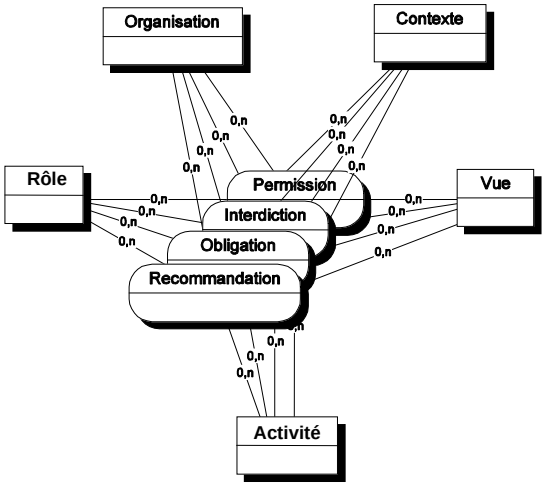

Figure 7 : Les relations *Permission*, *Interdiction*, *Obligation* et *Recommandation*.

Par exemple, la politique de sécurité de l’hôpital Purpan peut comporter les faits suivants : *Permission*(*Purpan*, *médecin*, *consulter*, *dossier_médical*, *urgence*) qui signifie que “l’hôpital Purpan accorde aux médecins la permission de consulter n’importe quel dossier médical dans le contexte de l’urgence” et *Permission*(*Purpan*, *médecin*, *consulter*, *dossier_médical*, *médecin_traitant*) qui signifie que “l’hôpital Purpan accorde aux médecins la permission de consulter les dossiers médicaux des patients dont ils sont les médecins traitants”.

4.8 Les autorisations concrètes

La relation *Permission* 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, nous introduisons dans notre modèle la relation *Est_permis* entre les sujets, les objets et les actions : si *s* est un sujet, α est une action et *o* est un objet, alors *Est_permis*(*s*, α, *o*) signifie que le sujet *s* a la permission de réaliser l’action α sur l’objet *o*. Les relations *Est_interdit*, *Est_obligatoire* et *Est_recommandé* sont définies de la même manière. Notre relation *Est_permis* est similaire à la notion de permission évoquée dans le modèle HRU (voir section 1). Il y a tout de même une différence de taille : dans le modèle HRU, les triplets d’autorisation <*s*, α, *o*> doivent être explicitement décrit ; alors que dans notre modèle, 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*. Ceci est modélisé par une règle générale présentée dans la section 5. Les instances explicites de la relation *Est_permis* peuvent être considérées comme des exceptions à la politique de sécurité générale spécifiée par la relation *Permission*.

La figure 8 résume notre modèle de sécurité. Il contient huit entités (*Organisation*, *Sujet*, *Rôle*, *Objet*, *Vue*, *Action*, *Activité* et *Contexte*) et douze relations (*Habilite*, *Utilise*, *Considère*, *Permission*, *Interdiction*, *Obligation*, *Recommandation*, *Est_permis*, *Est_interdit*, *Est_obligatoire*, *Est_Recommandé* et *Définit*).

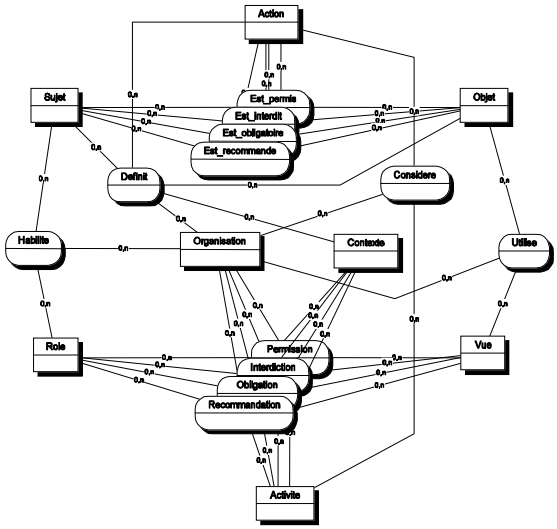

Figure 8 : le modèle ORBAC

5 Langage et axiomatique

L’objet de cette section est de présenter un cadre logique qui nous offre des outils d’aide au raisonnement sur les permissions, les interdictions, les obligations et les recommandations. L’objectif est d’associer un langage du premier ordre $\mathcal{L}$ au

diagramme entité-relation décrit précédemment. Notre langage du premier ordre doit fournir une syntaxe permettant d'exprimer les instances des relations existantes entre les entités. Chaque expression de $\mathcal{L}$ contiendra des symboles extraits d'un vocabulaire particulier classés en quatre groupes : les symboles de constante, les variables individuelles, les symboles de relation et les symboles de fonction.

Les symboles de constante correspondent aux instances des entités du diagramme. Ainsi, il y aura autant de type θ de symboles de constante que d'entités dans notre diagramme, c'est-à-dire huit : *Organisation*, *Sujet*, *Objet*, *Action*, *Rôle*, *Vue*, *Activité* et *Contexte*. Nous aurons par exemple les symboles de constante de type *Organisation* comme *Purpan*, *Rangueil*, *ICU31*, etc., les symboles de constante de type *Sujet* comme *Jean*, *Marie*, *ICU31*, etc, les symboles de constante de type *Objet* comme *F31.doc*, *F32.doc*, *F33.tex*, etc., les symboles de constante de type *Action* comme *lire*, *écrire*, *consulter*, etc., les symboles de constante de type *Rôle* comme *médecin*, *infirmière*, *unité_des_soins_intensifs*, etc., les symboles de constante de type *Vue* comme *dossier_administratif*, *dossier_médical*, *dossier_chirurgical*, etc., les symboles de constante de type *Activité* comme *lecture*, *écriture*, *consultation*, etc. Les constantes seront notées par des lettres minuscules comme *a*, *b* et *c*.

De la même manière, il y aura des variables individuelles pour chaque type θ. Elles seront notées par des lettres latines minuscules comme *x*, *y* et *z*. Pour tous les types θ, les symboles de constante de type θ et les variables individuelles de type θ seront appelés termes-θ.

Les symboles de relation de $\mathcal{L}$, notées par des lettres majuscules *P*, *Q*, *R*, etc., correspondront aux douze relations de notre diagramme. Chaque symbole de relation *P* de $\mathcal{L}$ est considéré comme un type de relation. Plus précisément, *Habilite* est un symbole de relation de type (*Organisation*, *Sujet*, *Rôle*). *Permission*, *Interdiction*, *Obligation* et *Recommandation* sont symboles de relation de type (*Organisation*, *Rôle*, *Vue*, *Activité*, *Contexte*). *Est_permis*, *Est_interdit*, *Est_obligatoire*, *Est_recommandé* sont des symboles de relation de type (*Sujet*, *Objet*, *Action*).

En utilisant les termes et les relations, nous construisons les formules atomiques de $\mathcal{L}$ comme suit : si t_1 est un terme-θ_1, …, t_n est un terme-θ_n et *P* est une relation de type (θ_1, …, θ_n), alors $P(t_1, \ldots, t_n)$ est une formule atomique. *Habilite*(*Purpan*, *Jean*, *médecin*) et *Permission*(*Rangueil*, *secrétaire_médicale*, *création*, *dossier_administratif*, *normal*) sont des formules atomiques.

A ce stade, notre langage n'est pas assez expressif pour pouvoir comparer des entités. Dans de nombreuses applications, nous désirons dériver des informations concernant certaines propriétés des entités. D'un point de vue formel, des symboles de fonction seront utilisés pour décrire les attributs de ces entités. Les symboles de fonction seront notés par des lettres minuscules comme *f*, *g* et *h*. A chaque symbole de fonction *f* sont associés un domaine et un co-domaine (encore appelé domaine image de la fonction). Le domaine et le co-domaine d'un symbole de fonction dépendent de la nature des attributs qui lui sont associés. Si un symbole de fonction correspond à l'attribut *Nom*, alors son domaine est de type *Sujet* et son co-domaine est un ensemble de noms. De même, le domaine d'un symbole de fonction correspondant à un attribut *Age* sera de type *Sujet* et son co-domaine sera un ensemble d'entiers positifs. Enfin, le domaine d'un symbole de fonction correspondant à l'attribut *Groupe_sanguin* sera de type *Sujet* et son co-domaine {*A*, *AB*, *B*, *O*}. Dans la mesure où il est possible que des sujets n'aient pas de nom ou que leur groupe sanguin soit inconnu, les symboles de fonction du langage $\mathcal{L}$ pourront n'établir qu'une correspondance partielle entre les domaines et co-domaines associés. Dans de nombreuses situations, il nous est impossible d'attribuer une valeur unique à certains attributs d'une entité. Pour répondre à une telle situation d'un point de vue conceptuel, nous utiliserons des symboles de fonction unaires ayant pour co-domaine l'ensemble des parties d'un ensemble (*Power Set*). Pour illustrer ceci, il nous suffit de mentionner le cas de l'attribut *médecin_traitant* : le domaine du symbole de fonction associé est de type *Sujet* et le co-domaine associé est un ensemble d'ensembles finis de noms.

Afin de dériver les informations représentées par les symboles de fonction, nous devons introduire des relations binaires concrètes, notées par σ, τ et μ entre les domaines. Le type d'une relation binaire concrète est le couple correspondant aux domaines sur lesquels la relation s'applique. L'égalité est probablement la relation binaire concrète la plus simple que nous aurons à traiter. Considérons les exemples suivants :

- Si *t* et *u* sont des termes de type *Sujet*, alors *médecin_traitant(t) = médecin_traitant(u)* signifie que les sujets *t* et *u* ont les mêmes médecins traitants,
- Si *t* et *u* sont des termes de type *Sujet*, alors *Âge(t) = Âge(u)* signifie que les sujets *t* et u ont le même âge et
- Si *t* et *u* sont des termes de type *Sujet*, alors *Groupe_sanguin(t) = Groupe_sanguin(u)* signifie que les sujets *t* et *u* ont le même groupe sanguin.

Bien évidemment, il existe certains cas où d'autres relations binaires doivent être considérées. Par exemple :

- Si *t* et *u* sont des termes de type *Sujet*, alors *médecin_traitant(t)* ∩ *médecin_traitant(u)* ≠ ∅ signifie que les sujets *t* et *u* ont un médecin traitant en commun,
- si *t* et *u* sont des termes de type *Sujet*, alors *Âge(t)* < *Âge(u)* signifie que le sujet *t* est plus jeune que le sujet *u* et
- si *t* et *u* sont des termes de type *Sujet*, alors *Groupe_sanguin(t)* ~ *Groupe_sanguin(u)* signifie que les groupes sanguins de *t* et *u* sont compatibles.

Ces types de formules seront aussi considérés comme des formules atomiques. Ainsi les formules de $\mathcal{L}$ sont définies comme suit :

- Une formule atomique est une formule,
- si *A* est une formule alors ¬*A* "non *A*" est une formule,
- si *A* et *B* sont des formules alors (*A* ∧ *B*) "*A* et *B*" et (*A* ∨ *B*) "*A* ou *B*" sont des formules et
- si *A* est une formule et *x* est une variable individuelle alors ∀*xA* "pour toutes les valeurs possibles de *x*, on a *A*" et ∃*xA* "il existe des valeurs possibles de *x* telles que *A*" sont des formules.

Comme d'habitude, les connecteurs logiques → et ↔ sont définis de la façon suivante : (*A* → *B*) est équivalent à (¬*A* ∨ *B*), et (*A* ↔ *B*) est équivalent à ((*A* → *B*) ∧ (*B* → *A*)). Nous omettrons volontairement les parenthèses quand aucune ambiguïté ne sera possible. Voici deux exemples de formules :

- ∀*s* (*Habilite*(*Rangueil*, *s*, *médecin*) → *Habilite*(*Rangueil*, *s*, *personnel_soignant*)) : "tous les médecins de l'hôpital Rangueil sont aussi habilités comme personnel soignant" et
- ∀*r*∀*v*∀*a* (*Permission*(*Rangueil*, *r*, *a*, *v*, *médecin_traitant*) → *Permission*(*Rangueil*, *r*, *a*, *v*, *urgence*)) "si l'hôpital Rangueil accorde au rôle *r* la permission de réaliser l'activité *a* sur la vue *v* dans le contexte "*médecin_traitant*", alors il accorde aussi cette permission dans le contexte d'urgence.

La valeur de vérité d'une formule est déterminée par les valeurs de ses sous formules dans un modèle donné. Les modèles pour notre langage consistent en huit ensembles non vides correspondant aux huit entités de notre diagramme et en douze relations correspondant aux douze relations de notre diagramme. Comme nous utilisons dans la suite la définition classique de la valeur de vérité d'une formule, il ne nous semble pas nécessaire de la rappeler ici.

Les axiomes d'une théorie du premier ordre sont habituellement divisés en axiomes logiques et axiomes propres. Les axiomes logiques fournissent les bases pour démontrer tous les théorèmes de la logique classique du premier ordre alors que les axiomes propres correspondent à des règles particulières. Nous supposons que toutes les politiques de sécurité seront basées sur la liste suivante d'axiomes propres pour toutes les organisations *org* :

1. ∀*s*∀*α*∀*o*∀*r*∀*a*∀*v*∀*c*
   Permission(*org*, *r*, *a*, *v*, *c*) ∧
   Habilite(*org*, *s*, *r*) ∧
   Utilise(*org*, *o*, *v*) ∧
   Considère(*org*, *α*, *a*) ∧
   Définit(*org*, *s*, *α*, *o*, *c*) → *Est_permis*(*s*, *α*, *o*) : "si l'organisation *org*, dans le contexte *c*, accorde la permission au rôle *r* de réaliser l'activité *a* sur la vue *v*, si *org* habilite le sujet *s* dans le rôle *r*, si *org* utilise l'objet *o* dans la vue *v*, si *org* considère l'action *α* comme faisant partie de l'activité *a* et si au sein de l'organisation *org* le contexte *c* est vraie entre *s*, *α* et *o*, alors le sujet *s* a la permission de réaliser l'action *α* sur l'objet *o*",
2. ∀*r*∀*a*∀*v*∀*c* (*Obligation*(*org*, *r*, *a*, *v*, *c*) → *Recommandation*(*org*, *r*, *a*, *v*, *c*) : "toutes les obligations sont aussi des recommandations",
3. ∀*r*∀*a*∀*v*∀*c* (*Recommandation*(*org*, *r*, *a*, *v*, *c*) → *Permission*(*org*, *r*, *a*, *v*, *c*) : "toutes les recommandations sont aussi des permissions",
4. ∀*r*∀*a*∀*v*∀*c* (*Permission*(*org*, *r*, *a*, *v*, *c*) → ¬*Interdiction*(*org*, *r*, *a*, *v*, *c*) : "une permission implique une non interdiction".

L'axiome 1 décrit comment des permissions abstraites entre des rôles, des vues et des activités peuvent être transformées en permissions concrètes entre des sujets, des objets et des actions. Les axiomes pour les obligations, les recommandations et les interdictions sont définis de la même manière.

6 Exemple de politique de sécurité

Dans cette section nous montrons comment exprimer un exemple simple de politique de sécurité dans notre langage.

6.1 Les sujets et les rôles

Dans cet exemple, nous ne considérons que l'organisation "hôpital Purpan" (figure 9). Nous supposons que l'hôpital Purpan habilite plusieurs sujets : Jean dans le rôle de directeur, Marie dans le rôle d'assistante administrative, ST1 dans le rôle

d'équipe chirurgicale et RT2 dans le rôle d'équipe radiologique. Dans notre langage, ces faits sont représentés par des instances de la relation *Habilite* :

- *Habilite*(*Purpan*, *Jean*, *directeur*),
- *Habilite*(*Purpan*, *Marie*, *assistante_administrative*),
- *Habilite*(*Purpan*, *ST1*, *équipe_chirurgicale*) et
- *Habilite*(*Purpan*, *RT2*, *équipe_radiologique*).

Dans ces faits, *directeur*, *assistante_administrative*, *équipe_chirurgicale* et *équipe_radiologique* sont des rôles.

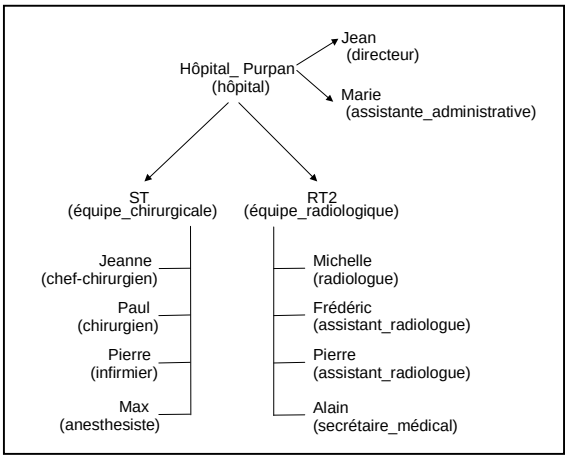

Figure 9 : Exemple d'organisation

La sous organisation ST1 habilite d'autres sujets : Jeanne dans le rôle de chef de l'équipe chirurgicale, Paul dans le rôle de chirurgien, Pierre dans le rôle d'infirmier et Max dans celui d'anesthésiste. Nous obtenons ainsi de nouvelles instances de la relation *Habilite* :

- *Habilite*(*ST1*, *Jeanne*, *chef_chirurgien*),
- *Habilite*(*ST1*, *Paul*, *chirurgien*),
- *Habilite*(*ST1*, *Pierre*, *infirmier*) et
- *Habilite*(*ST1*, *Max*, *anesthésiste*).

De la même manière, des sujets sont habilités par RT2, l'équipe radiologique. De plus, nous considérons qu'un attribut *Patient* associé à l'entité *Sujet* indique quels sont les patients d'un sujet. Par conséquent, notre langage inclut une fonction partielle *Patient* ayant pour domaine *Sujet* et pour co-domaine un ensemble d'ensembles finis de noms. Par exemple, les fonctions *Patient(Purpan)* et *Patient(Michelle)* retournent respectivement la liste des patients de l'hôpital Purpan et de Michelle.

6.2 Les objets et les vues

Considérons les objets appartenant aux vues suivantes :

- *dossier_administratif* : Les objets qui appartiennent à cette vue fournissent des informations administratives concernant les patients comme leur nom, leur adresse, leur âge, etc,
- *dossier_médical* : Cette vue correspond au dossier médical des patients et,
- *dossier_chirurgical* : Cette vue correspond aux dossiers confidentiels gérés par l'équipe chirurgicale.

Nous supposons que les objets appartenant à ces vues ont un attribut *Nom*. Ainsi, si *F31.doc* est un dossier appartenant à une de ces vues, alors *Nom*(*F31.doc*) fournit le nom du patient correspondant. Nous supposons également que les dossiers sont directement gérés par l'hôpital Purpan, dans une base de données relationnelle par exemple. Ceci se traduit dans notre modèle par des faits de la forme :

- *Utilise*(*Purpan*, *F31.doc*, *dossier_administratif*),
- *Utilise*(*Purpan*, *F32.doc*, *dossier_médical*) et
- *Utilise*(*Purpan*, *F33.tex*, *dossier_chirurgical*).

ST1, l'équipe chirurgicale et RT2, l'équipe radiologique, partagent la même base de données gérée par l'hôpital Purpan. Cela signifie qu'elles utilisent les mêmes vues que l'hôpital. Ceci peut s'exprimer de la manière suivante :

- $\forall o \forall v$ (*Utilise*(*Purpan*, *o*, *v*) $\rightarrow$ *Utilise*(*ST1*, *o*, *v*)),
- $\forall o \forall v$ (*Utilise*(*Purpan*, *o*, *v*) $\rightarrow$ *Utilise*(*RT2*, *o*, *v*)).

A partir des trois vues *dossier_administratif*, *dossier_médical*, *dossier_chirurgical*, nous définissons une quatrième vue, appelée *dossier_patient*. Nous supposons qu'elle a trois attributs *dossier_administratif*, *dossier_médical* et *dossier_chirurgical* tels que :

- $\forall o$ (*Utilise*(*Purpan*, *o*, *dossier_patient*) $\leftrightarrow$
  $\exists o_1 \exists o_2 \exists o_3$
  (*Utilise*(*Purpan*, o_1, *dossier_administratif*) $\wedge$
  Utilise(*Purpan*, o_2, *dossier_médical*) $\wedge$
  Utilise(*Purpan*, o_3, *dossier_chirurgical*) $\wedge$
  dossier_administratif(o) = o1 $\wedge$
  dossier_médical(o) = o2 $\wedge$
  dossier_chirurgical(o) = o3 $\wedge$
  Nom(o1) = Nom(o2) = Nom(o3)).

La vue *dossier_patient* correspond au dossier médical complet du patient. Dans une base de données relationnelle, nous obtiendrions le dossier patient par une jointure des vues *dossier_administratif*, *dossier_médical*, *dossier_chirurgical* suivant l'attribut *Nom*.

6.3 Les actions et les activités

Nous ne considérons ici que les activités correspondant à des accès directs aux dossiers, c'est-à-dire la création, la consultation et l'écriture, etc. Si nous supposons que les dossiers sont gérés par l'hôpital dans une base de données relationnelle, ces activités correspondent respectivement aux actions *insert*, *select*, *update*, etc. Ceci est représenté par les trois faits suivants :

- *Considère*(*Purpan*, *select*, *creation*),
- *Considère*(*Purpan*, *select*, *consulter*) et
- *Considère*(*Purpan*, *update*, *écriture*).

6.4 Les contextes

Nous modélisons trois contextes dans notre exemple : "médecin traitant", "équipe traitante" et "urgence". Le contexte "médecin traitant" est défini dans ST1, l'équipe chirurgicale, comme suit :

- $\forall s \forall o \forall \alpha$ (*Définit*(*ST1*, *s*, α, *o*, *médecin_traitant*) $\leftrightarrow$ *Nom*(*o*) $\in$ *Patient*(*s*)) : dans ST1, le contexte "médecin traitant" est vrai entre le sujet *s*, l'action α et l'objet *o* si et seulement si *s* joue un rôle dans ST1 et si *o* est un dossier correspondant à un patient traité par le sujet *s*.

Le contexte "équipe traitante" est défini de la manière suivante :

- $\forall s \forall o \forall \alpha$ (*Définit*(*ST1*, *s*, α, *o*, *equipe_traitante*) $\leftrightarrow$ $\exists r$ (*Habilite*(*ST1*, *s*, *r*) $\wedge$ *Nom*(*o*) $\in$ *Patient*(*ST1*)) : dans ST1, le contexte "équipe traitante" est vrai entre le sujet *s*, l'action α et l'objet *o* si et seulement si *s* joue un rôle dans ST1 et si *o* est un dossier correspondant à un des patients traité par l'organisation ST1.

De même, le contexte "urgence" est définit comme suit :

- $\forall s \forall o \forall \alpha$ (*Définit*(*ST1*, *s*, α, *o*, *urgence*) $\leftrightarrow$ vrai): dans ST1, le contexte "urgence" est toujours vrai entre les sujets, les actions et les objets.

6.5 La politique de sécurité

Nous ne pouvons dans cet article entrer dans le développement complet de la spécification d'une politique de sécurité d'un établissement hospitalier. Nous ne présentons que quelques exemples illustrant comment une telle politique de sécurité peut être exprimée à l'aide de notre modèle. Considérons les trois permissions suivantes :

- *Permission*(*RT2*, *médecin*, *consulter*, *dossier_médical*, *medecin_traitant*),
- *Permission*(*RT2*, *médecin*, *consulter*, *dossier_médical*, *équipe_traitante*) et
- *Permission*(*RT2*, *médecin*, *consulter*, dossier_*chirurgical*, *équipe_traitante*).

La première permission indique que l'organisation RT2 permet aux médecins de consulter les dossiers médicaux des patients dont ils sont les médecins traitants. Les deuxième et troisième permissions spécifient que l'organisation RT2 permet aux médecins de consulter un dossier médical ou chirurgical si ce dossier correspond à un patient de RT2. Notons que les permissions associées aux rôles médecin peuvent changer d'une organisation à une autre, et les contextes respectifs sont aussi susceptibles d'être différents. Cette possibilité est en particulier intéressante dans notre exemple dans la mesure où les conditions dans lesquelles un médecin à la permission de consulter un dossier peut varier selon s'il exerce dans une équipe chirurgicale ou une équipe radiologique.

6.6 Les hiérarchies

Jusqu'à présent nous n'avons pas abordé la notion de hiérarchie de rôles. Cette notion à d'abord été introduite dans le modèle RBAC [15]. L'idée est de mettre en place un mécanisme d'héritage des permissions à travers la hiérarchie de rôle. Dans notre approche, la hiérarchie de rôle n'est pas considérée comme un concept de base. L'héritage des permissions dans ST1 entre un rôle r_1 (par exemple médecin) et un rôle r_2 (par exemple chirurgien) est spécifié par la règle suivante :

- $\forall a \forall v \forall c$ (*Permission*(*ST1*, r_1, *a*, *v*, *c*) $\rightarrow$ *Permission*(*ST1*, r_2, *a*, *v*, *c*).

Nous pouvons ajouter à notre langage une relation *Sous-rôle*(*ST1*, r_1, r_2). Les instances d'une telle relation étant simplement définies par la règle que nous venons de définir. Précisons toutefois que dans notre modèle il est possible de spécifier que l'héritage entre deux rôles donnés ne s'applique qu'à certaines organisations. Nous pouvons par exemple spécifier qu'à l'hôpital Purpan, le rôle directeur hérite des permissions du rôle médecin. :

- $\forall a \forall v \forall c$ (*Permission*(*Purpan*, *médecin*, *a*, *v*, *c*) $\rightarrow$ *Permission*(*Purpan*, *directeur*, *a*, *v*, *c*)).

Bien sur, nous n'aurions pas cette règle si, au lieu de considérer l'hôpital Purpan, nous considérons un autre établissement au sein duquel le directeur n'est pas un médecin. Il est également possible d'exprimer dans notre modèle l'héritage, entre rôles, d'interdictions, d'obligations et de recommandations. Nous pouvons également une hiérarchie entre les vues et considérer que les permissions sont héritées à travers cette hiérarchie. Le même principe peut être appliqué à une hiérarchie entre les activités. Par exemple, les vues *dossier_administratif*, *dossier_médical* et *dossier_chirurgical* sont des sous vues de la vue *dossier_patient*. Ainsi, un rôle qui a la permission de réaliser une activité sur la vue *dossier_patient*, a également la permission de réaliser la même

activité sur les sous vues précédemment citées. Ceci s'exprime dans notre langage par la règle suivante :

- $\forall r \forall a \forall c$ (*Permission*(*Purpan, r, a, dossier_patient, c*) $\rightarrow$ *Permission*(*Purpan, r, a, dossier_administratif, c*)).

Il en est de même pour les vues *dossier_médical* et *dossier _chirurgical*.

7 Les contraintes

L'utilisation de contraintes a été proposée dans le modèle RBAC [15]. Les contraintes sont exprimées dans notre modèle par des règles s'appliquant à diverses relations. Nous donnons ici quelques exemples :

- $\forall s$ (*Habilite*(*Purpan, s, équipe_chirurgicale*) $\rightarrow$ ($\exists s_1$ *Habilite*(s, s_1, *chirurgien*) $\wedge$ $\exists s_2$ *Habilite*(s, s_2, *anesthésiste*) $\wedge$ $\exists s_3$ *Habilite*(s, s_3, *infirmier*)) : Cette règle indique que si l'hôpital Purpan habilite *s* comme équipe chirurgicale, alors *s* habilite un chirurgien, un anesthésiste et une infirmière.
- $\forall s$ $\neg$(*Habilite*(*Purpan, s, chirurgien*) $\wedge$ *Habilite*(*Purpan, s, anesthésiste*)) : Au sein de l'hôpital Purpan, aucun sujet *s* ne peut être habilité à la fois comme chirurgien et comme anesthésiste.
- $\forall s \forall s'$ *Habilite*(*Purpan, s, directeur*) $\wedge$ *Habilite*(*Purpan, s', directeur*) $\rightarrow$ $s = s'$: Au sein de l'hôpital Purpan, un seul sujet *s* peut être employé comme directeur.

Il est ainsi possible d'exprimer de nombreuses contraintes sur les relations *Utilise*, *Considère*, *Définit*, *Permission*, *Obligation*, *Interdiction*, *Prohibition* ou *Recommandation*.

8 Conclusion

Dans cet article, nous avons présenté un nouveau modèle de politique de sécurité dont le but est d'apporter des réponses aux limites des modèles existants. Ce modèle, appelé ORBAC, est centré sur le concept d'organisation. En effet, tous les autres concepts que nous avons définis et qui permettent de spécifier une politique de sécurité dépendent d'une organisation donnée :

- Le concept de *Rôle* permet de modéliser comment l'organisation habilite les sujets,
- le concept de *Vue* permet de modéliser comment l'organisation utilise les objets,
- le concept d'*Activité* permet de modéliser comment l'organisation réalise des actions, et
- la relation *Définit* permet de modéliser comment l'organisation définit des contextes dans lesquels des utilisateurs réalisent des actions sur des objets.

En s'appuyant sur ces concepts, nous pouvons exprimer une politique de sécurité comme un ensemble de permissions, d'interdictions, d'obligations et de recommandations. Une permission correspond à un fait ayant la forme *Permission*(*org, r, a, v, c*) qui signifie que l'organisation *org*, dans le contexte *c*, permet au rôle *r* de réaliser l'activité *a* sur la vue *v*. Les faits *Interdiction*, *Obligation* et *Recommandation* sont définis de façon analogue. Nous avons également montré comment dériver des permissions, des interdictions, des obligations et des recommandations concrètes qui s'appliquent à des sujets, des actions et des objets.

Plusieurs problèmes n'ont pas été abordés dans cet article. Tout d'abord, des conflits peuvent apparaître dans la politique de sécurité. Par exemple, pour un sujet donné, une action donnée, et un objet donné, il nous faut détecter et résoudre une situation dans laquelle il serait possible de dériver à la fois une permission et une interdiction. Cette question a fait l'objet de diverses propositions [4, 10, 18]. L'approche que nous suggérons s'appuie sur la logique possibiliste, et doit permettre de dériver automatiquement des niveaux de priorité entre les règles de la politique de sécurité [3]. Si nous n'avons pas discuté ici de la question de l'administration de la politique de sécurité, il est bien évidemment nécessaire de développer un modèle complet incorporant l'administration. Un tel modèle a été présenté avec ARBAC [16] pour l'administration du modèle RBAC. Le modèle d'administration de notre modèle sera abordé dans un futur article. Enfin, il nous faut aussi montrer comment spécifier des propriétés de sécurité dans le modèle ORBAC. En particulier, nous devons définir des moyens pour détecter une violation de la politique de sécurité et spécifier la décision à prendre dans un tel cas ; comme par exemple dans le cas d'un sujet qui ne remplirait pas ses obligations.

Remerciements

Le travail présenté dans cet article a été effectué dans le cadre du projet RNRT MP6 (***M***odèles et ***P***olitiques de ***S***écurité des ***S***ystèmes d'***I***nformations et de ***C***ommunication en ***S***anté et en ***S***ocial).

Bibliographie

[1] J. Barkley, K. Beznosoz et J. Uppal. Supporting Relationships in Access Control Using Role Based Access Control. *Proceeding of the ACM workshop on RBAC*, Fairfax, Virginia, USA, 28-29 Octobre 1999.

[2] D. E. Bell et L. J. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Technical Report ESD-TR-73-306, The MITRE Corporation, Mars 1976.

[3] S. Benferhat, R. El Baida et F. Cuppens. Modélisation des politiques de sécurité dans le cadre de la théorie des possibilités. *Rencontres Francophones de la Logique Floue et ses Applications*, Montpellier, France, Octobre 2002.

[4] E. Bertino, S. Jajodia et P. Samarati. Supporting Multiple Access Control Policies in Database Systems. *IEEE Symposium on Security and Privacy*, Oakland, USA, 1996.

[5] C. Bettini, S. Jajodia, X. S. Wang et D. Wijesekera. Obligation Monitoring in Policy Management. *International Workshop, Policies for Distributed Systems and Neworks (Policy 2002)*, Monterey CA, 5-7 Juin 2002.

[6] K. J. Biba. Integrity consideration for secure computer systems. Technical Report MTR-3153, The MITRE Corporation, Juin 1975.

[7] E. C. Cheng. An Object-Oriented Organizational Model to Support Dynamic Role-based Access Control in Electronic Commerce Applications. *32nd Annual Hawaii International Conference on System Sciences (HICSS-32)*, Maui, Hawaii, 5-8 Janvier 1999.

[8] F. Cuppens, L. Cholvy, C. Saurel et J. Carrère. Merging Regulations: analysis of a practical example. *International Journal of Intelligent Systems*, 16(11), Novembre 2001.

[9] N. Damianou, N. Dulay, E. Lupu et M. Sloman. The Ponder Policy Specification Language. *International Workshop, Policies for Distributed Systems and Neworks (Policy 2001)*. Bristol, UK, 29-31 Janvier 2001.

[10] G. Dinolt, L. Benzinger et M. Yatabe. Combining Components and Policies. *Proceedings of the Computer Security Foundations Workshop VII*, Franconia, USA, 1994.

[11] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn et R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, 4(3):222-274, Août 2001.

[12] S. I. Gavrila et J. F. Barkley. Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management. *Third ACM Workshop on Role-Based Access Control*, pages 81-90, 22-23 Octobre 1996.

[13] M. A. Harrison, W. L. Ruzzo et J. D. Ullman. Protection in Operating Systems. *Communication of the ACM*, 19(8):461-471, Août 1976.

[14] B. Lampson. Protection. *5th Princeton Symposium on Information Sciences and Systems*, pages 437-443, Mars 1971.

[15] R. Sandhu, E. J. Coyne, H. L. Feinstein et C.E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38-47, 1996.

[16] Ravi Sandhu, Bhamidipati et Qamar Munawer. The ARBAC97 Model for Role-Based Administration of Roles. *ACM Transactions on Information and System Security*, 2(1), Février 1999.

[17] R. Thomas et R. Sandhu. Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management. *11th IFIP Working Conference on Database Security*, Lake Tahoe, California, USA, 1997.

[18] Roshan K. Thomas. TMAC: A primitive for Applying RBAC in collaborative environment. *2nd ACM, Workshop on RBAC*, pages 13-19, Fairfax, Virginia, USA, 6-7 Novembre 1997.