

# ÉVALUATION ET COMPARAISON DES MODÈLES DE CONTRÔLE D'ACCÈS

par

Herman Pooda

Mémoire présenté au Département d'informatique  
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES  
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 11 décembre 2015



# Sommaire

La protection des données et de la vie privée des personnes est devenue aujourd'hui un enjeu majeur pour les entreprises et les organisations gouvernementales qui collectent et entreposent les données à caractère personnel. L'adoption d'une politique de sécurité est un impératif. Plusieurs modèles de contrôle d'accès sont proposés dans la littérature pour guider les utilisateurs dans la mise en oeuvre de leurs politiques de sécurité. Chacun de ces modèles a ses forces et faiblesses. Les systèmes de contrôle d'accès mis en place s'érigent souvent en de véritables obstacles, rendant inefficace le travail de leurs utilisateurs. Il convient de mieux connaître les modèles de contrôles d'accès avant de les implémenter. Ce mémoire présente une étude complète des modèles de contrôle d'accès RBAC, XACML et SGAC, en dégageant les enjeux auxquels les utilisateurs devront s'attendre en adoptant ces modèles. RBAC et XACML ont été normalisés respectivement par ANSI et OASIS et sont actuellement les modèles dominants dans l'industrie et dans le monde de la recherche. SGAC est un modèle proposé à la suite d'une étude, pour implémenter le contrôle d'accès aux dossiers médicaux au profit du réseau de santé de Sherbrooke. Les récentes études ont montré que c'est dans le domaine de la santé que les violations de la vie privée sont plus fréquentes. Le mémoire présente aussi les principales exigences d'un système de contrôle d'accès dans le domaine de la santé. Sur la base des exigences identifiées, il propose une évaluation des modèles de contrôle d'accès étudiés, et fournit une comparaison de ces modèles.

**Mots-clés:** contrôle d'accès ; dossier santé électronique ; protection de la vie privée ; évaluation des modèles de contrôle d'accès ; comparaison des modèles de contrôle d'accès.

## SOMMAIRE

# Remerciements

Tout d’abord, je voudrais exprimer ma gratitude à l’Agence Canadienne de Développement International qui, à travers le Programme Canadien des Bourses de la Francophonie (PCBF), m’a permis de faire cette étude.

J’exprime ma gratitude au Professeur Marc Frappier de l’Université de Sherbrooke, de m’avoir accepté dans son laboratoire, et de tout l’encadrement qu’il m’a offert. J’adresse également mes remerciements à Mohammed Ouenzar, pour le suivi dont j’ai bénéficié de sa part durant les travaux.

Je remercie mes collègues étudiants du laboratoire GRIL (Groupe de Recherche en Ingénierie du Logiciel), pour le soutien dont j’ai bénéficié de leur part.

Je n’oublie pas ma famille et mes amis qui, de près ou de loin, n’ont jamais cessé de m’encourager et de me soutenir dans leur prière.

## REMERCIEMENTS

# Abréviations

<b>ABAC</b>	Attribute-Based Access Control
<b>ANSI</b>	American National Standards Institute
<b>CHUS</b>	Centre Hospitalier Universitaire de Sherbrooke
<b>DAC</b>	Discretionary Access Control
<b>DME</b>	Dossier Médical Electronique
<b>DSE</b>	Dossier Santé Electronique
<b>GTRBAC</b>	Generalized Temporal Role-Based Access Control
<b>MAC</b>	Mandatory Access Control
<b>NIST</b>	National Institute of Standards and Technology
<b>OASIS</b>	Organization for the Advancement of Structured Information Standards
<b>PAP</b>	Policy Administration Point
<b>PDP</b>	Policy Decision Point
<b>PEP</b>	Policy Enforcement Point
<b>PIP</b>	Policy Information Point
<b>P-RBAC</b>	Privacy-aware Role-Based Access Control
<b>RBAC</b>	Role-Based Access Control
<b>SGAC</b>	Solution de Gestion des Accès et du Consentement
<b>URI</b>	Uniform Resource Identifier
<b>XACML</b>	eXtensible Access Control Markup Language

## ABRÉVIATIONS



# Table des matières

<b>Sommaire</b>	<b>iii</b>
<b>Remerciements</b>	<b>v</b>
<b>Abréviations</b>	<b>vii</b>
<b>Table des matières</b>	<b>ix</b>
<b>Liste des figures</b>	<b>xiii</b>
<b>Liste des tableaux</b>	<b>xv</b>
<b>Liste des programmes</b>	<b>xvii</b>
<b>Introduction</b>	<b>1</b>
<b>1 État de l'art</b>	<b>5</b>
1.1 Définition des concepts . . . . .	5
1.2 Modèles de contrôle d'accès . . . . .	7
1.2.1 Les modèles de contrôle d'accès discrétionnaire (DAC : Discretionary Access Control) . . . . .	8
1.2.2 Les modèles de contrôle d'accès mandataire (MAC : Mandatory Access Control ) . . . . .	10
1.2.3 Les modèles de contrôle d'accès basés sur les rôles (RBAC : Role Based Access Control) . . . . .	14
1.3 Contrôle d'accès aux dossiers de santé électronique (DSE) . . . . .	15
	ix

## TABLE DES MATIÈRES

1.4	Evaluation des modèles de contrôle d'accès . . . . .	20
<b>2</b>	<b>Cas d'étude</b>	<b>23</b>
2.1	Architecture du système . . . . .	23
2.2	Exigences du système . . . . .	24
2.3	Besoins du système . . . . .	26
<b>3</b>	<b>Application de RBAC au cas d'étude</b>	<b>29</b>
3.1	Présentation de RBAC . . . . .	29
3.2	Application de RBAC au cas d'étude . . . . .	34
3.2.1	Scénario 1 . . . . .	35
3.2.2	Scénario 2 . . . . .	36
3.2.3	Scénario 3 . . . . .	37
3.2.4	Scénario 4 . . . . .	40
3.2.5	Scénario 5 . . . . .	40
3.2.6	Scénario 6 . . . . .	43
3.2.7	Scénario 7 . . . . .	44
3.3	Critiques RBAC . . . . .	45
<b>4</b>	<b>Application de XACML au cas d'étude</b>	<b>49</b>
4.1	Présentation de XACML . . . . .	50
4.1.1	Langage XACML . . . . .	50
4.1.2	Architecture du modèle de contrôle d'accès XACML . . . . .	55
4.1.3	Profils XACML utilisés pour modéliser le cas d'étude . . . . .	56
4.1.4	Processus de prise de décision . . . . .	58
4.2	Mise en oeuvre de XACML . . . . .	61
4.2.1	Balana . . . . .	61
4.2.2	Fonctionnement du système implementé . . . . .	62
4.2.3	Conclusion de la mise en oeuvre de XACML . . . . .	66
4.3	Modélisation du cas d'étude en XACML . . . . .	67
4.3.1	Représentation des ressources dans notre système . . . . .	70
4.3.2	Scénario 1 . . . . .	71
4.3.3	Scénario 2 . . . . .	72

## TABLE DES MATIÈRES

4.3.4	Scénario 3 . . . . .	75
4.3.5	Scénario 4 . . . . .	75
4.3.6	Scénario 5 . . . . .	77
4.3.7	Scénario 6 . . . . .	79
4.3.8	Scénario 7 . . . . .	81
4.4	Critiques XACML . . . . .	82
<b>5</b>	<b>Application de SGAC au cas d'étude</b>	<b>85</b>
5.1	Présentation du modèle SGAC . . . . .	85
5.1.1	Expression des règles en SGAC . . . . .	86
5.1.2	Évaluation des requêtes . . . . .	86
5.2	Modélisation du cas d'étude . . . . .	91
5.2.1	Scénario 1 . . . . .	91
5.2.2	Scénario 2 . . . . .	93
5.2.3	Scénario 3 . . . . .	94
5.2.4	Scénario 4 . . . . .	95
5.2.5	Scénario 5 . . . . .	96
5.2.6	Scénario 6 . . . . .	97
5.2.7	Scénario 7 . . . . .	97
5.3	Critiques de SGAC . . . . .	99
<b>6</b>	<b>Identification des critères d'évaluation et comparaison de RBAC, XACML et SGAC</b>	<b>101</b>
6.1	Critères d'évaluation . . . . .	101
6.1.1	Expression des politiques . . . . .	103
6.1.2	Administration des politiques . . . . .	103
6.1.3	Autoriser les exceptions ( <i>Break the glass</i> ) . . . . .	104
6.1.4	Principe de moindre privilège . . . . .	105
6.1.5	Délégation des privilèges . . . . .	106
6.1.6	Expression des conditions . . . . .	107
6.1.7	Contrainte de séparation des tâches . . . . .	107
6.1.8	Mécanisme de resolution de conflits . . . . .	107
6.1.9	Prise en compte des interdictions et des permissions . . . . .	108

## TABLE DES MATIÈRES

6.1.10 Performance . . . . .	108
6.1.11 Expression des obligations . . . . .	114
6.2 Résumé de la comparaison de RBAC, XACML et SGAC . . . . .	114
<b>Conclusion</b>	<b>117</b>
<b>A Code du PEP et du Context Handler</b>	<b>119</b>
<b>B Code du PIP</b>	<b>123</b>

# Liste des figures

1.1	Relations entre la politique, le modèle et le mécanisme de contrôle d'accès	8
1.2	Matrice d'autorisation du modèle de Lampson . . . . .	9
1.3	Pas de lecture en haut et pas d'écriture en bas . . . . .	11
1.4	Gestion des conflits d'intérêt de la Muraille de Chine . . . . .	13
1.5	Architecture du Réseau de santé de Walloom . . . . .	18
1.6	Modèle P-RBAC . . . . .	18
2.1	Architecture du système de contrôle d'accès du CHUS . . . . .	25
2.2	Graphe des sujets . . . . .	26
2.3	Graphe des types de ressources . . . . .	27
3.1	Flux d'information dans un système RBAC . . . . .	31
3.2	Modèle entités-associations du core RBAC . . . . .	31
3.3	Héritage entre les rôles dans hiérarchie RBAC . . . . .	32
3.4	Modélisation de la séparation statique des tâches en RBAC . . . . .	33
3.5	Modélisation de la séparation dynamique des tâches en RBAC . . . . .	34
3.6	Accès accordé à tout le personnel de l'hôpital sur les données de Patrice	35
3.7	Retrait des autorisations d'accès aux données de Patrice . . . . .	36
3.8	Autorisation d'accès aux laboratoires de Romain . . . . .	38
3.9	Restriction des autorisations d'accès aux laboratoires de Romain à un groupe d'infirmières . . . . .	39
3.10	Modélisation du scénario 4 . . . . .	41
3.11	Modélisation de $E_{5.1}$ . . . . .	42
3.12	Modélisation de $E_{5.1}$ et $E_{5.2}$ . . . . .	42

## LISTE DES FIGURES

3.13	Modélisation du scénario 6 . . . . .	44
3.14	Modélisation du scénario 7 sans le contexte temps . . . . .	45
4.1	Modèle du langage XACML, tel que fourni dans la version 3 de la norme XACML . . . . .	55
4.2	Codes simplifiés des politiques $P_1$ , $P_2$ et $P_3$ . . . . .	59
4.3	Diagramme de flux de données du système XACML . . . . .	67
4.4	Modèle logique de données de la base de données du système XACML . . . . .	68
5.1	Hiérarchie des sujets . . . . .	87
5.2	Graphe des types de ressources . . . . .	88
5.3	Graphe des types de ressources instancié avec les ressources de Simon . . . . .	89
5.4	Graphe des types de ressources instancié avec les ressources du patient Patrice . . . . .	92
6.1	Comparaison entre XACML et SGAC, suivant le temps moyen d'exécution d'une requête après chargement des règles et des graphes . . . . .	114

# Liste des tableaux

3.1	Description des figures utilisées pour représenter les scénarios . . . . .	34
5.1	Modélisation du scénario 1 . . . . .	92
5.2	Modélisation du scénario 2 . . . . .	94
5.3	Modélisation du scénario 3 . . . . .	95
5.4	Modélisation du scénario 4 . . . . .	96
5.5	Modélisation du scénario 5 . . . . .	97
5.6	Modélisation du scénario 6 . . . . .	97
5.7	Modélisation du scénario 7 . . . . .	98
5.8	Résumé de la modélisation de l'ensemble des scénarios . . . . .	98
6.1	Résultats des tests de performances des systèmes XACML et SGAC .	113
6.2	Comparaison entre RBAC, XACML et SGAC . . . . .	115

## LISTE DES TABLEAUX



# Liste des programmes

4.1	Modélisation de l'exemple1 en XACML . . . . .	51
4.2	Format XACML de la requête d'Alice Fertier . . . . .	64
4.3	Format XACML de la requête d'Alice Fertier réécrite . . . . .	65
4.4	Exemple de réponse XACM . . . . .	66
4.5	Modélisation de la cible du scénario 1 . . . . .	72
4.6	Politique contenant les règles de la loi . . . . .	73
4.7	Modélisation exigence 2 du scénario 2 . . . . .	74
4.8	cible du scenario 4 . . . . .	76
4.9	Modélisation de l'exigence 1 du scénario 4 . . . . .	77
4.10	Modélisation de l'exigence 1 du scénario 5 . . . . .	78
4.11	Modélisation de l'exigence 2 du scénario 5 . . . . .	79
4.12	Modélisation du scénario 6 . . . . .	80
4.13	Modélisation du scénario 7 . . . . .	82

## LISTE DES PROGRAMMES

# Introduction

## Contexte

La gestion électronique de l'information a des avantages indéniables pour les entreprises, mais elle pose également un grand défi ; car si l'information n'est pas bien protégée, elle peut entraîner des conséquences très néfastes pour l'entreprise. Dans le domaine de la santé, les hôpitaux ont basculé dans la gestion électronique des dossiers médicaux, et la tendance actuelle est de migrer vers un système universel de gestion électronique des dossiers médicaux. Plus les systèmes s'ouvrent, plus les risques de violation des données sont grands. Plusieurs modèles standards de contrôle d'accès existent dans la littérature, pour aider les entreprises dans la mise en oeuvre des systèmes pour contrôler les accès aux ressources protégées. La multiplication des modèles de contrôle d'accès est due au fait qu'aucun des modèles actuellement présents dans la littérature, ne permet de couvrir tous les besoins qu'on peut rencontrer dans les entreprises en matière de contrôle d'accès.

Il ressort des modèles standards qui ont été adoptés dans l'industrie, que beaucoup de systèmes issus de l'implémentation de ces modèles, deviennent des obstacles [47, 24], en empêchant les utilisateurs d'exercer correctement leurs fonctions. Pour contourner les obstacles, les utilisateurs implémentent de nouvelles fonctionnalités à côté de celles décrites dans le modèle. L'ajout de nouvelles fonctionnalités peut entraîner des failles pour le système.

La protection des données et la vie privée demeure une question à laquelle il faut trouver une solution. En cette année 2015, les sondages sur les violations des données ont peint un tableau très noir, car les impacts sur le plan économique et social, sont énormes [31].

## Problématique

Le problème de violation de la vie privée est toujours d'actualité malgré l'existence des solutions. Les principales causes sont :

1. les modèles standards n'arrivent pas à couvrir les besoins et les exigences du contrôle d'accès ;
2. les modèles standards ont des limites qui ne sont pas bien connues par les utilisateurs.

## Objectifs

Au regard des problèmes soulevés dans la section problématique, cette étude se fixe les objectifs suivants :

### **A court terme :**

1. identifier les limites des modèles existants ;
2. cerner toutes les exigences en matière de contrôle d'accès, dans le domaine de la gestion électronique des dossiers médicaux ;
3. établir une comparaison entre les modèles de contrôle d'accès, afin de déterminer celui qui sied le mieux pour implémenter le contrôle d'accès dans le domaine de la santé.

### **A long terme :**

4. proposer un cadre générique pouvant permettre d'implémenter les politiques de contrôle d'accès dans n'importe quel domaine d'affaires.

## Méthodologie

Les exigences et les besoins du contrôle d'accès dans le domaine de la santé, sont très diversifiés. Nous sommes convaincus que, si dans ce domaine, on a un modèle

## INTRODUCTION

sûr, qui couvre l'ensemble des besoins et exigences, on pourra à partir de ce modèle, élaborer un modèle plus générique qui pourra s'appliquer dans n'importe quel domaine d'affaires. La méthode pour atteindre l'objectif à long terme, consiste donc, à partir d'un cas particulier pour ensuite le généraliser. Pour atteindre les objectifs à court terme précédemment énumérés, il est bon d'avoir une bonne maîtrise des modèles existants et, de connaître les besoins et exigences en matière de contrôle d'accès. Ainsi, le travail commence par une étude des modèles de contrôle d'accès standards. Est associé à cette étude, un modèle privé qui a été proposé à la suite d'une étude, comme solution pour implémenter le contrôle d'accès au profit du réseau de santé de la région de Sherbrooke.

Nous avons aussi exploré les systèmes de contrôle d'accès mis en place par certaines organisations, le but étant de déterminer les besoins et les exigences du contrôle d'accès. Des lois gouvernementales, des directives d'organismes de réglementation oeuvrant dans le domaine de la protection de la vie privée, des travaux de recherches ont également été consultés pour enrichir les listes des exigences et des besoins du contrôle d'accès.

Après le travail d'identification des exigences et besoins, les modèles sont évalués suivant des critères spécifiés. L'évaluation faite dans cette étude n'est pas seulement théorique. Pour certains critères, l'évaluation est faite sur des systèmes implémentant les modèles de contrôle d'accès.

## Résultats

Le travail a conduit à :

1. élucider un certain nombre de questions auxquelles les utilisateurs devront s'attendre en adoptant les modèles standards ;
2. à identifier les limites des modèles de contrôle d'accès standards ;
3. à identifier les exigences du contrôle d'accès des systèmes de gestion électronique des dossiers médicaux ;
4. à établir une comparaison des modèles de contrôle d'accès sur la base des exigences identifiées.

# Structure du mémoire

Le mémoire est organisé en 6 chapitres.

Le chapitre 1 présente une revue de littérature portant sur les modèles de contrôles d'accès, sur l'utilisation des modèles de contrôle d'accès dans le domaine de santé et sur les travaux d'évaluation et de comparaison des modèles de contrôle d'accès.

Le chapitre 2 présente un cas d'étude, tiré du réseau de santé de la région de Sherbrooke. Ce cas d'étude est utilisé tout au long du travail.

Les chapitres 3, 4 et 5 présentent respectivement les études des modèles RBAC, XACML et SGAC à travers leur application au cas d'étude.

Le chapitre 6 présente les exigences du contrôle d'accès dans le domaine de la santé, évalue les modèles RBAC , XACML et SGAC et établit une comparaison entre les trois modèles.

Une conclusion clôt le mémoire.

# Chapitre 1

## État de l’art

Ce chapitre présente les modèles de contrôle d’accès qui existent dans la littérature. Il présente ensuite l’utilisation des modèles de contrôle d’accès dans la protection des dossiers de santé électronique (DSE). Sont également concernés par cette revue de littérature, les travaux précédant le nôtre, et relatifs à l’évaluation et à la comparaison des modèles de contrôle d’accès.

### 1.1 Définition des concepts

**Dossier de santé électronique :** «On appelle dossier de santé électronique (DSE) les systèmes qui forment le dossier à vie, sécurisé et confidentiel où figurent les antécédents d’une personne en matière de santé et de soins. Ces systèmes entreposent et transmettent des renseignements médicaux comme les résultats d’analyses de laboratoire, les profils pharmacologiques, les principaux rapports cliniques (p. ex. : sommaires de congés), l’imagerie diagnostique (p. ex. : radiographies) et les vaccins reçus. Les professionnels de la santé autorisés ont accès à cette information par voie électronique.» [30]

**Dossier médical électronique :** « Le dossier médical électronique (DME) est un système bureautique qui permet aux professionnels de la santé, comme les médecins de

## CHAPITRE 1. ÉTAT DE L'ART

famille, d'enregistrer l'information recueillie lors d'une consultation. Cette information, par exemple le poids du patient, sa tension artérielle et d'autres renseignements cliniques, aurait auparavant été consignée à la main et conservée dans une chemise au cabinet du médecin. Bientôt, le DME permettra aussi au médecin d'accéder au dossier de santé complet d'un patient, y compris à l'information du DSE qui a été recueillie par d'autres professionnels de la santé. » [29]

**Dossier de santé personnel :** « dossier de santé partiel ou complet sous la garde d'une ou de plusieurs personnes (p. ex. un patient ou un membre de sa famille) qui renferme une partie ou l'intégralité des informations sur la santé de cette personne durant toute sa vie. Il s'agit aussi d'un dossier centré sur une personne, mais contrairement au DSE, c'est le patient plutôt que le professionnel de la santé qui exerce un contrôle sur ce dossier ou en est le gardien. » [29]

**Rôle :** le rôle désigne une fonction dans une organisation, exemple : médecin, infirmier dans un hôpital. Le rôle regroupe un ensemble de privilèges et les octroie aux utilisateurs qui lui sont associés.

**Sujet :** c'est une entité active, généralement une personne, un processus ou un dispositif qui provoque la circulation de l'information dans un système, ou change l'état d'un système [36].

**Objet / ressource :** c'est une entité passive qui contient ou reçoit des informations. L'accès à un objet implique potentiellement l'accès à l'information qu'il contient. Exemples d'objets : les enregistrements dans une base de données, les pages web, les fichiers, les répertoires, les programmes, les imprimantes, le noeud d'un réseau informatique [36].

**Opération :** c'est un processus actif appelé par un sujet. Par exemple quand un utilisateur entre sa carte et un numéro d'identification valide dans un guichet automatique de banque, plusieurs choix d'opérations lui sont offerts. L'utilisateur peut décider de faire un dépôt, ou un retrait ou une demande de solde [20].



## 1.2. MODÈLES DE CONTRÔLE D'ACCÈS

**Permission (privilège) :** c'est une autorisation d'effectuer une opération sur le système. Dans la plupart des documents sur la sécurité informatique, le terme permission se réfère à une combinaison entre un objet et une opération. Par exemple, lire la fiche d'examen d'un patient.

## 1.2 Modèles de contrôle d'accès

Le contrôle d'accès détermine qui (sujet) a accès à quoi (objet) et dans quelle condition. La mise en place d'un système de contrôle d'accès suit une approche qui se déroule en plusieurs étapes nommées par les concepts suivants : **politique de contrôle d'accès**, **modèle de contrôle d'accès** et **mécanisme de contrôle d'accès** [48].

La politique de contrôle d'accès définit les règles de haut niveau qui déterminent les droits d'accès aux objets. La politique est mise en œuvre par un mécanisme qui évalue les requêtes des utilisateurs et rend une décision. Le mécanisme implémente les fonctionnalités spécifiées dans la politique et représentées formellement dans le modèle de contrôle d'accès. Le modèle de contrôle d'accès est une représentation formelle de la politique de contrôle d'accès et de son fonctionnement.

La figure 1.1 montre les relations entre la politique de contrôle d'accès, le modèle de contrôle d'accès et le mécanisme de contrôle d'accès. Un système de contrôle d'accès est d'abord initié par une politique de contrôle d'accès, puis formalisé avec un modèle de contrôle d'accès. La politique de contrôle d'accès est enfin mise en œuvre directement ou indirectement (à travers le modèle) par un mécanisme matériel ou logiciel.

Dans la littérature, il n'y a pas une classification formelle des politiques de contrôle d'accès. Certains auteurs classent les politiques de contrôle d'accès en deux catégories : **les politiques discrétionnaires** et les **politiques non-discrétionnaires** [28]. Les politiques de contrôle d'accès discrétionnaires sont associées à l'identité du sujet, et les privilèges d'accès aux objets sont gérés par le sujet lui-même. Les politiques non-discrétionnaires sont basées sur des règles administrées par une autorité centrale.

Les récents travaux sur la sécurité des systèmes d'information classent les politiques de contrôle d'accès en trois catégories [48] : **Les politiques discrétionnaires**, les

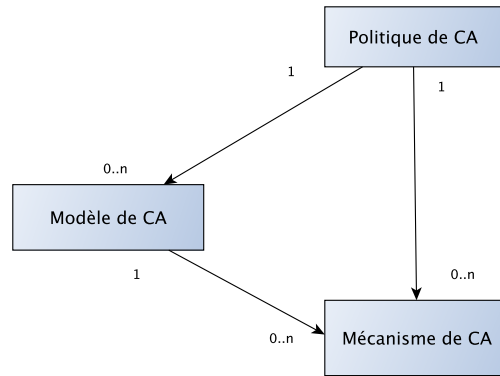


figure 1.1 – Relations entre la politique, le modèle et le mécanisme de contrôle d'accès

**politiques mandataires (obligatoires selon certaines appellations) et les politiques basées sur les rôles.** Cette classification scinde la classe des politiques non-discrétionnaires en deux : les politiques mandataires et les politiques basées sur les rôles. La particularité des politiques basées sur les rôles, réside dans le fait que, dans ces politiques, les privilèges d'accès aux objets ne sont jamais affectés directement au sujet, mais indirectement à travers le rôle auquel le sujet est assigné.

Les sous-sections qui suivent, présentent les modèles de contrôle d'accès associés à chaque catégorie de politiques.

### 1.2.1 Les modèles de contrôle d'accès discrétionnaire (DAC : Discretionary Access Control)

Dans les modèles DAC, l'utilisateur qui crée un objet, a tous les droits sur cet objet. Il contrôle les accès des autres utilisateurs. La gestion des accès aux fichiers du système d'exploitation UNIX, constitue un exemple classique du mécanisme du contrôle d'accès discrétionnaire. Le premier modèle discrétionnaire est proposé par Lampson [38].

Lampson a introduit la notion de matrice d'autorisation (figure 1.2). Les droits d'accès aux objets sont représentés à travers un triplet  $(S, O, M)$  où  $S$  désigne l'ensemble des sujets,  $O$  l'ensemble des objets et  $M$  la matrice d'autorisation. Chaque cellule  $M[s, o]$  de la matrice contient les droits d'accès que possède le sujet  $s$  sur l'objet

## 1.2. MODÈLES DE CONTRÔLE D'ACCÈS

*o.* Les droits d'accès sont : *lire (read)*, *écrire (write)*, *exécuter (execute)*, *contrôler (control)* et *posséder (own)*.

	Fichier 1	Fichier 2	Fichier 3	Programme 1
Anne	Own read write	read write		execute
Bob	read		read write	
Alice		read		execute

figure 1.2 – Matrice d'autorisation du modèle de Lampson

Le modèle de Lampson a été redéfini par Graham et Denning [23]. Graham et Denning ont défini des règles pour créer, supprimer, transférer et donner les permissions d'accès, dans le but de mettre à jour la matrice d'autorisation.

Les modèles DAC sont connus pour être intrinsèquement limités pour deux raisons. Premièrement, l'octroi d'un droit d'accès est transitif; par exemple, quand Anne accorde à Bob un accès à un fichier, rien n'empêche Bob de copier le fichier de Anne dans un autre dossier que Bob contrôle. Bob peut maintenant accorder à un autre utilisateur, l'accès à la copie du fichier de Anne à l'insu de Anne. Deuxièmement, les modèles DAC sont vulnérables aux attaques de type cheval de Troie [28]. Par exemple, Bob peut écrire un programme pour Anne qui à priori effectue une fonction utile, tandis que dans le même temps détruit le contenu des fichiers de Anne. Après vérification, on trouvera que Anne a détruit ses propres fichiers.

Les insuffisances des modèles DAC sont :

- l'information peut être copiée d'un objet à un autre; par conséquent, il n'y a pas de véritable assurance sur la circulation de l'information dans un système.
- les privilèges d'accès à un objet sont décidés par le propriétaire de l'objet, plutôt que par une politique de l'ensemble du système qui reflète les exigences de sécurité de l'organisation.

Les modèles de contrôle d'accès mandataires ont été proposés pour apporter des solutions aux insuffisances des modèles discrétionnaires.

### 1.2.2 Les modèles de contrôle d'accès mandataire (MAC : Mandatory Access Control )

Afin de remédier au problème de fuites d'information des modèles discrétionnaires, les modèles mandataires fixent des règles incontournables destinées à forcer le respect des exigences du contrôle d'accès. Ce n'est pas l'utilisateur qui définit les règles lui-même. Les modèles mandataires diffèrent selon qu'il s'agit de maintenir la confidentialité ou l'intégrité des données. Le premier modèle, appelé modèle de Bell-LaPadula [17], a été développé pour le département de la défense américaine et vise, plus particulièrement à assurer la confidentialité. Le deuxième modèle, appelé modèle de Biba [32], s'est intéressé à l'intégrité. D'autres modèles mandataires comme le modèle Muraille de Chine [12], ont été développés pour les systèmes commerciaux.

#### Modèle de Bell-LaPadula

Le modèle de Bell-Lapadula a pour objectif de contrôler la confidentialité des données. Dans ce modèle, les sujets et les objets sont classés suivant des niveaux de sécurité (*top secret*, *secret*, *public*), comme le montre la figure 1.3. Pour un objet, le niveau de sécurité représente le danger que peut constituer la divulgation de l'information contenue dans cet objet. Pour un sujet, le niveau de sécurité représente le niveau de confiance qui lui est accordé. Le contrôle d'accès est imposé par deux principes :

1. **Pas de lecture en haut** : un sujet peut lire un objet si le niveau de sécurité du sujet domine celui de l'objet. Le niveau de sécurité du sujet domine celui de l'objet si le niveau de sécurité du sujet est supérieur à celui de l'objet. Par exemple le niveau de sécurité (*top secret*) est supérieur au niveau de sécurité (*secret*).
2. **Pas d'écriture en bas** : un sujet peut écrire dans un objet si le niveau de sécurité de l'objet domine celui du sujet. Ce qui permet d'éviter qu'un sujet de niveau supérieur de mettre de l'information secrète dans un objet de niveau inférieur auquel ont accès les sujets de niveau inférieur.

Pour la formulation de leur modèle Bell et Lapadula se sont basés sur deux théorèmes appelés en anglais «*Basic Security Theorem (BST)*», qui stipulent que, un

## 1.2. MODÈLES DE CONTRÔLE D'ACCÈS

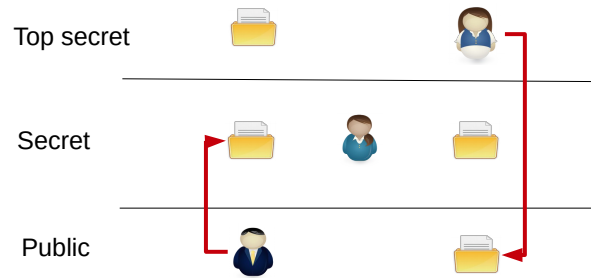


figure 1.3 – Pas de lecture en haut et pas d'écriture en bas

système est sécurisé si :

- i. son état initial  $v_0$  est sécurisé, et
- ii. l'état  $v_0$  est préservé par la transition  $T$  qui transforme un état sécurisé en un autre état sécurisé.

Un système se compose d'un état initial  $v_0$ , d'un ensemble de requêtes  $R$ , et d'une fonction  $T : VXR \rightarrow V$  qui transforme un état du système dans un autre état résultant de l'exécution d'une requête. Une des faiblesses du modèle de Bell-Lapadula présentée par Mclean [41], réside dans le fait qu'il n'y a pas de contrainte placée sur la transition  $T$ . Un sujet de niveau inférieur peut changer le niveau de sécurité des sujets et des objets de niveau supérieur, et permettre l'accès à tous les sujets. L'écriture dans un niveau dominant étant permis, dès lors que le niveau de sécurité est mémorisé dans l'objet, le sujet de niveau inférieur peut alors modifier le niveau de sécurité.

Autres limites du modèle de Bell-Lapadula :

- le modèle est très rigide et difficile à utiliser dans la pratique : l'information circule dans un seul sens : du bas vers le haut. Dans les entreprises où l'information circule dans tous les sens, l'adoption de ce modèle va entraver l'exécution des tâches ;
- le modèle ne garantit pas l'intégrité des données.

### Modèle de Biba

Le modèle de Biba met l'accent sur la protection de l'intégrité des données. Les sujets et les objets sont classés par niveau de sécurité comme dans le modèle de

Bell-Lapadula. L'information la plus intègre est celle du niveau le plus haut. Pour maintenir cet état, le modèle de Biba se base sur les deux principes suivants :

1. **Pas de lecture en bas** : le sujet  $s$  peut lire l'objet  $o$  si le niveau de sécurité de l'objet domine celui du sujet. Ce qui permet d'éviter qu'un sujet de niveau supérieur ne prenne connaissance d'informations moins intègres que celles du niveau auquel il appartient.
2. **Pas d'écriture en haut** : le sujet  $s$  peut écrire dans l'objet  $o$  si le niveau de sécurité du sujet domine celui de l'objet.

L'information ne doit pas circuler d'un bas niveau à un niveau élevé. Ceci assure que l'information n'est pas contaminée avec de l'information moins intègre. Pour permettre un contrôle dynamique des deux principes ci-dessus, Biba propose deux règles :

1. concernant le principe *Pas de lecture en bas*, aucune restriction n'est imposée à l'écriture. Cependant, si un sujet  $s$  écrit sur un objet  $o$ , l'objet  $o$  est déclassé à la borne inférieure de la classification de  $s$  et  $o$ .
2. concernant le principe *Pas d'écriture en haut*, aucune restriction n'est imposée à la lecture. Cependant, quand un sujet  $s$  lit un objet  $o$ , le sujet  $s$  est déclassé à la borne inférieure de la classification de  $s$  et  $o$ .

Le modèle de Biba a une limite majeure : la dégradation du niveau d'intégrité. L'idée de migrer un sujet vers un niveau d'intégrité inférieure lorsqu'il accède à un objet de niveau inférieur, va renvoyer tous les sujets en bas de l'échelle. Ce qui remet en cause l'intégrité du modèle puisqu'il n'y a plus de différence entre les sujets.

### Modèle Muraille de Chine

Le modèle Muraille de Chine est inspiré du commerce. Le but de ce modèle est de prévenir le flux d'informations entre des clients concurrents. Par exemple, une compagnie  $C$  a différents clients parmi lesquels  $A$  et  $B$  sont en compétition entre eux. La compagnie  $C$  veut s'assurer qu'aucune information à propos de  $A$  n'est donnée à  $B$ , et vice versa. Le but visé par le modèle est atteint par imposition de deux règles :

1. Règle 1 : La lecture de l'objet  $o$  est accordée à un sujet  $s$  seulement :

## 1.2. MODÈLES DE CONTRÔLE D'ACCÈS

- (a) si  $o$  appartient au client pour lequel l'accès est requis,
  - (b) ou, si  $o$  appartient à un autre client qui n'est pas en compétition avec le client pour lequel l'accès est requis. Dans la figure 1.4, le sujet Bob qui a accès aux objets du client  $A$  ne peut plus avoir accès aux objets du client  $B$ . Bob peut avoir accès aux objets de  $C$  ou (exclusif) aux objets de  $D$ .
2. Règle 2 : un sujet  $s$  est autorisé à écrire dans un objet  $o$ ,
- (a) si l'accès à  $o$  est autorisé par la règle 1
  - (b) et, si  $s$  ne peut lire aucun objet  $o2$  tel que : i)  $o2$  appartient à un client différent du propriétaire de  $o$ , et ii)  $o2$  contient une information non aseptisée (une information aseptisée est une information dépourvue des détails sensibles et qui n'est sujette à aucune restriction d'accès).

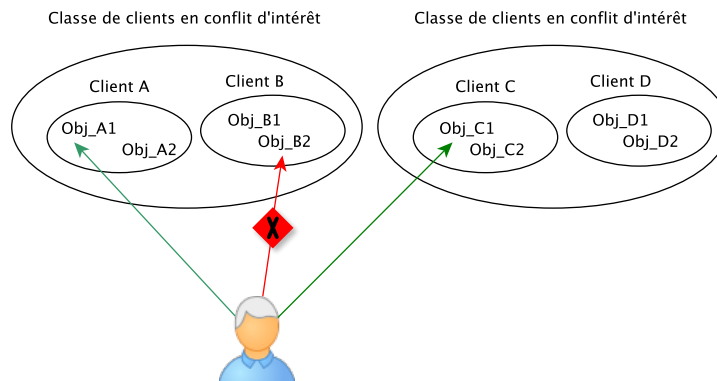


figure 1.4 – Gestion des conflits d'intérêt de la Muraille de Chine

Les limites du modèle Muraille de Chine sont liées à la gestion de l'historique. L'historique des accès est utilisé dans l'évaluation des requêtes. Lorsque l'historique n'est pas conservé le système n'est plus sûr. Le modèle est aussi très restrictif et difficile à utiliser dans un système réel comme le démontre Sandhu [49].

### 1.2.3 Les modèles de contrôle d'accès basés sur les rôles (RBAC : Role Based Access Control)

RBAC [22] est une norme de l'ANSI. Il est actuellement le modèle prédominant pour le contrôle d'accès. RBAC est recommandé par Inforoute Santé du Canada [30], une société créée par les provinces canadiennes et le gouvernement fédéral pour promouvoir le développement des systèmes d'information de santé au Canada. Dans RBAC, les autorisations sont affectées à des rôles et les utilisateurs gagnent les autorisations en étant assignés aux rôles. Le rôle dans RBAC désigne une fonction qu'occupe un agent au sein d'une entreprise. RBAC respecte la structure organisationnelle des entreprises. Les rôles peuvent être structurés de façon hiérarchique, permettant ainsi l'héritage des autorisations. Les séparations statiques et dynamiques des tâches sont également prises en compte dans ce modèle. RBAC a été largement étudié et étendu dans la littérature. Parmi ses extensions, RB-RBAC [4, 42] ajoute les autorisations négatives. La délégation d'autorisations est considérée dans [15, 59]. P-RBAC [44] prend en compte les obligations. GTRBAC [11] considère les contraintes temporelles sur les règles de contrôle d'accès. ABAC [33, 37, 18] généralise RBAC en ajoutant des prédicats sur les attributs rôles et ressources, qui peuvent être utilisés pour définir les conditions d'application des règles. XACML [50] est une norme de OASIS qui fournit un langage dédié aux modèles ABAC. XACML fournit également une architecture, permettant d'implémenter un système de contrôle d'accès.

Malgré des décennies de recherche, le problème de contrôle d'accès n'a pas encore été résolu de manière satisfaisante. Les principaux défis d'aujourd'hui sont l'intégration et l'élargissement. La tendance actuelle, est l'orientation vers les modèles ABAC comme XACML, qui sont des systèmes apatrides. RBAC et XACML sont des modèles choisis pour cette étude. Nous reviendrons en détails sur ces deux modèles dans les chapitres 3 et 4. La section suivante présente l'utilisation des modèles de contrôle d'accès dans la vraie vie, plus précisément dans le domaine des systèmes de santé électronique.



## 1.3 Contrôle d'accès aux dossiers de santé électronique (DSE)

Les DSE contribuent fortement à améliorer la qualité des soins et à réduire les dépenses dans le domaine de la santé. La tendance actuelle est la centralisation par région des dossiers de santé électronique, dans une perspective d'aboutir à un DSE universel. Déjà, la Commission européenne veut stimuler l'économie numérique en permettant à tous les Européens d'avoir accès aux dossiers médicaux électroniques partout en Europe d'ici 2020 [35]. La centralisation des dossiers médicaux soulève une préoccupation majeure concernant la protection de la vie privée des patients.

Les données médicales sont très sensibles. Leur violation expose le patient à des problèmes économiques, à des angoisses mentales, à une stigmatisation par la société, etc. La violation des données médicales est classée au premier rang des violations des données électroniques selon l'étude «*2015 Cost of Data Breach Study*» [31]. Les DSE sont exposés à plusieurs menaces. Deux catégories de menaces selon [7] :

- *les menaces organisationnelles* liées aux agents internes du système qui abusent de leurs privilèges ou bien aux agents externes qui exploitent les failles du système.
- *Les menaces systémiques* liées aux agents du système qui utilisent les données à d'autres fins.

Au regard de la sensibilité des DSE et des impacts de leur violation, des textes législatifs ont été adoptés pour régir la mise en place des politiques de contrôle d'accès. La *Directive 95/46/EC* du parlement européen sur la protection des individus au regard des traitements effectués sur les données personnelles et la Directive américaine HIPAA («*federal Health Insurance Portability and Accountability Act*») qui porte plus particulièrement sur la protection des données de la santé, sont des exemples de ces textes. La mise en place d'un système de contrôle d'accès est devenu une obligation pour toute organisation de la santé qui met en place un DSE.

Les récentes revues de littérature ont montré que RBAC est le modèle de contrôle d'accès le plus largement utilisé dans les DSE [7, 6]. Cependant RBAC a des limites telles que l'impossibilité de gérer les événements imprévus ou dynamiques (transferts de patients, demandes d'avis entre médecins, etc). Ces limites font que les utilisateurs sont

obligés d'implémenter d'autres fonctionnalités ajoutées à RBAC afin de couvrir leurs besoins. Lillian Rostad et Ole Edsberg [47] ont montré que RBAC a été implémenté pour contrôler les accès aux dossiers médicaux des patients dans les organisations de santé de Norvège. Mais les utilisateurs ont vite contourné le système RBAC à cause des obstacles rencontrés dans l'utilisation. Le travail était devenu moins efficace avec le système RBAC. De nouvelles fonctionnalités ont été implémentées et ajoutées au système RBAC afin de contourner les obstacles rencontrés par les utilisateurs. Le constat de Lillian Rostad et Ole Edsberg se consolide avec ceux de Grandison et Davis [24]. L'ajout de nouvelles fonctionnalités crée des vulnérabilités pour le système [40]. Les fonctionnalités ajoutées n'étant pas présentes dans le modèle de base, elles n'ont pas été prises en compte dans la vérification des propriétés de sécurité avant l'implémentation du modèle. Plusieurs auteurs ont proposés des modèles qui étendent RBAC en lui ajoutant de nouveaux modules pour combler ses limites.

Parmi les extensions de RBAC, le modèle Privacy-awareness RBAC (P-RBAC) a été proposé, à l'issue des travaux de Annanda et Jean-Noël [46] comme celui qui convient le mieux pour protéger les dossiers médicaux dans le Réseau de Santé de Walloom (RSW). Le modèle P-RBAC (voir Figure 1.6) étend le modèle RBAC en ajoutant à RBAC de nouvelles entités pour prendre en compte le but, les conditions et les obligations. Le RSW a une architecture qui ressemble à celle de notre cas d'étude présenté au chapitre 2. Le RSW est un système de gestion électronique des dossiers médicaux des patients de la région de Walloom. Les principales caractéristiques de ce système sont :

- toutes les structures de santé (hôpitaux, clinique, etc.) de la région de Walloom sont interconnectées ;
- chaque structure de santé dispose de sa propre base de données contenant les dossiers de ses patients ;
- le système contient un serveur central, comme le montre la figure 1.5. Le serveur central comporte une base de données contenant toutes les règles de contrôle d'accès des patients, une base de données d'index pointant sur les données des patients et une base de données contenant des résumés des données de tous les patients ;
- un seul point de contrôle d'accès pour tout le système, qui reçoit et traite les

### 1.3. CONTRÔLE D'ACCÈS AUX DOSSIERS DE SANTÉ ÉLECTRONIQUE (DSE)

requêtes des utilisateurs ;

- la gestion des droits d'accès aux dossiers des patients est entièrement dévolue aux patients ou à des personnes désignées par les patients eux-mêmes. Le patient crée lui-même ses propres règles et peut offrir l'accès à n'importe quel praticien de l'hôpital. Le rôle de l'hôpital est mineur et se résume à assurer la sécurité physique des entrepôts de données ;
- cinq catégories d'utilisateurs peuvent accéder au dossier d'un patient : médecin généraliste, médecin spécialiste, médecin d'urgence, tuteur (exemple parent d'un mineur) et personne de confiance du patient (exemples mari, épouse) ;
- chaque utilisateur qui veut accéder au dossier d'un patient doit avoir une relation thérapeutique avec le patient (cette relation est définie par le patient). L'accès est autorisé sous condition du consentement du patient et doit répondre à un but spécifique : archive personnel, urgence, etc. ;
- chaque accès au dossier d'un patient est notifié au patient.

L'examen du cas d'étude de Walloom et de la solution qui a été proposée laissent planer des inquiétudes sur la sûreté du système à garantir la vie privée des patients. Ces inquiétudes sont :

- En délaissant la gestion des règles aux mains des patients, il y a un risque que les règles émanant de la réglementation ne soient pas prises en compte dans le système. En matière de gestion de la vie privée, il est fréquent que certaines règles proviennent des organismes de régulation. Ces genres de règles ont souvent préséance sur les règles définies par le patient. Il faut dans ce cas un administrateur qui puisse mettre à jour ces règles et s'assurer qu'elles sont vraiment présentes dans le système.
- Le modèle P-RBAC ne permet pas de représenter les interdictions. Nous verrons avec l'étude de RBAC au chapitre 3 qu'il y a un énorme risque de violation de la politique de sécurité dès lors que les règles d'interdiction ne peuvent pas être modélisées.

Pour choisir le modèle P-RBAC, Annanda et Jean-Noël ont effectué un travail d'évaluation en choisissant comme critères : la capacité d'expression des politiques, la performance dans l'évaluation des requêtes et la capacité à prendre en compte toutes les exigences du système. Annanda et Jean-Noël ont aussi mené un autre travail [45]

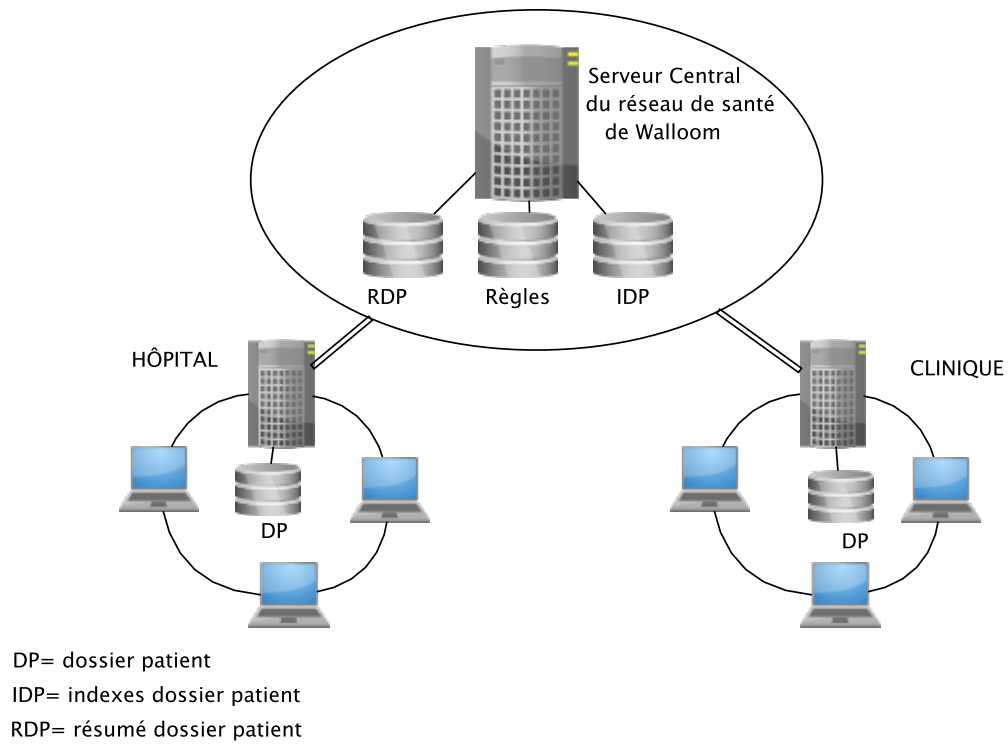


figure 1.5 – Architecture du Réseau de santé de Walloom

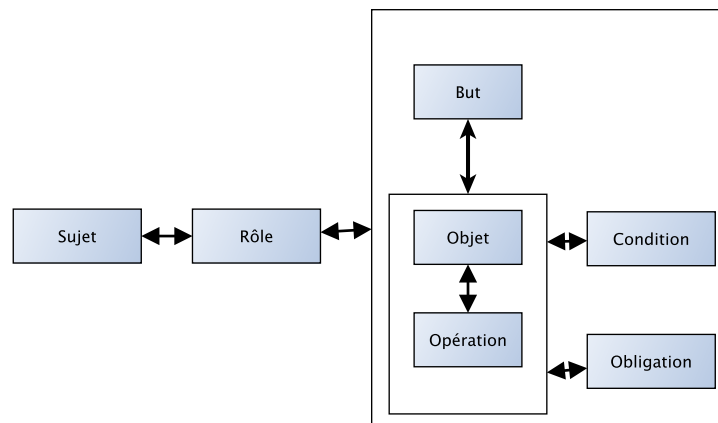


figure 1.6 – Modèle P-RBAC

### 1.3. CONTRÔLE D'ACCÈS AUX DOSSIERS DE SANTÉ ÉLECTRONIQUE (DSE)

qui a consisté à identifier les exigences pour le contrôle d'accès dans les systèmes de gestion électroniques des dossiers médicaux. Les exigences qu'ils ont identifiées résument à :

- la restriction des actions : il s'agit des actions comme imprimer, lire, écrire, transférer ; qu'un utilisateur peut effectuer sur les données d'un patient ;
- l'expression des contraintes temporelles et géographiques. Par exemple, accorder l'accès à un médecin pour une durée déterminée ;
- la prise en compte du principe du moindre privilège ;
- la possibilité de déléguer les permissions ;
- la prise en compte des obligations. Par exemple, envoyer un courriel au patient quand un utilisateur accède à son dossier ;
- la possibilité d'exprimer les buts des accès.

Les exigences identifiées par Annanda et Jean-Noël dans leur travail [45] sont inspirées de la Directive 95/46/EC du parlement européen sur la protection des individus au regard des traitements effectués sur les données personnelles. Nous nous sommes intéressés à ce travail, ainsi qu'à bien d'autres comme celui de InfoRoute Canada et HIPAA, pour cerner les principales exigences à considérer dans la mise en oeuvre d'un système de contrôle d'accès dans les DSE.

Outre RBAC, d'autres modèles de contrôle d'accès sont également utilisés dans le domaine de la santé. XACML définit le profil XSPA[53] (« *Cross-Enterprise Security and Privacy Authorization* ») qui spécifie l'utilisation de XACML 2.0 pour promouvoir l'interopérabilité au sein d'une communauté de soins de santé. XSPA fournit une sémantique et une nomenclature commune pour décrire les requêtes, les réponses aux requêtes, les politiques, la mise en application des politiques. XSPA harmonise HL7 [27] qui est un standard d'échange, d'intégration, de partage des informations électroniques dans le domaine de la santé. Axiomatic [10] démontre que XACML combiné avec le profil XSPA peut bien se substituer à HL7 dans un environnement où HL7 est déjà présent. Cependant pour avoir des politiques conformes à la réglementation dans le domaine de la santé, Axiomatic recommande l'utilisation de XACML seul. Pour appuyer cette assertion, Axiomatic prend l'exemple du système de santé suédois où XACML est utilisé pour gérer les autorisations d'accès. Omar Chowdhury et al. [14] ont utilisé XACML pour modéliser HIPAA. Ils se sont rendu compte que XACML

ne permettait pas de capturer toutes les règles définies dans HIPAA. En effet, avec XACML, il est impossible de modéliser certaines caractéristiques de HIPAA telles que les références entre les articles et les événements historiques. Ils ont alors proposé une extension de XACML pour permettre de modéliser entièrement HIPAA.

Le choix d'un modèle de contrôle d'accès se fait après une évaluation sur la base des exigences identifiées par l'utilisateur. La section suivante présente les travaux d'évaluation des modèles de contrôle d'accès rencontrés dans la littérature.

## 1.4 Evaluation des modèles de contrôle d'accès

Au regard du nombre élevé des modèles de contrôle d'accès, il convient de procéder d'abord à leur évaluation afin de déterminer celui qui convient le mieux à la politique de contrôle d'accès de l'utilisateur. Les risques sont grands quand le modèle n'a pas été bien choisi. Le système qui en découle peut s'ériger en obstacle pour l'utilisateur [47, 24], et lorsque les fonctionnalités manquantes sont ajoutées par l'utilisateur, cela rend le système vulnérable aux menaces de sécurité [40, 47]. Bien que ce travail préalable d'évaluation soit très important, peu d'auteurs se sont penchés dessus.

Le modèle de contrôle d'accès est une abstraction du système de contrôle d'accès. Il est utilisé pour vérifier théoriquement les propriétés de sécurité du système de contrôle d'accès. Le système implementé est à l'image du modèle qui est utilisé. Hasani et al. [40] ont identifié de façon générale, les principaux critères d'évaluation à considérer lors de la comparaison des modèles de contrôle d'accès. Les critères identifiés dans leurs travaux sont présentés comme les plus importants qui doivent être considérés dans l'évaluation des modèles. Dans notre travail, nous proposons d'identifier les critères de comparaison en fonction des besoins et des exigences du système à implémenter. C'est en fonction des besoins et des exigences du système qu'un critère de comparaison peut être qualifié d'important ou pas. Par exemple, le critère *performance* n'est pas cité par Hasani et al. alors que dans [45, 28], il est démontré que ce critère est très important dans les systèmes comportant plusieurs utilisateurs et dans le domaine de la santé.

L'évaluation sur la base du modèle de contrôle d'accès est jugée insuffisante par Hu *et al.* [28] qui proposent une évaluation plus large en considérant le système de contrôle d'accès en entier. Pour Hu *et al.* certaines propriétés du système comme

#### 1.4. EVALUATION DES MODÈLES DE CONTRÔLE D'ACCÈS

l'administration du système, ne peuvent pas être correctement vérifiées en utilisant juste le modèle. Les systèmes de contrôle d'accès comportent une grande variété de fonctionnalités et de capacités administratives, et l'impact opérationnel peut être important. En particulier, cet impact peut se rapporter à la productivité des utilisateurs, ainsi qu'à la capacité de l'organisation à accomplir sa mission. Par conséquent, il est raisonnable d'utiliser une mesure de qualité pour vérifier les capacités administratives, les coûts administratifs, la couverture de la politique, l'extensibilité, et les performances des systèmes de contrôle d'accès.

L'approche que nous avons adoptée dans le cadre de cette étude, prend en compte l'évaluation théorique sur la base du modèle de contrôle d'accès et l'évaluation pratique quand c'est nécessaire. Dans notre approche, nous identifions d'abord les critères d'évaluation en fonction des besoins et des exigences du système, puis nous classons les critères selon leur importance pour le système. Nous commençons l'évaluation en considérant d'abord le modèle de contrôle d'accès. Si pendant l'évaluation théorique il y a des propriétés importantes qui ne peuvent pas être vérifiées, nous faisons appel à l'évaluation pratique.

**Conclusion** Cette revue de littérature a permis de cerner les principaux enjeux à examiner lorsqu'on entreprend de mettre en place un système de contrôle d'accès. Nous avons pris connaissance des modèles standards, ce qui nous a permis d'écarter ceux qui présentent trop de limites, et de retenir deux modèles parmi les standards. Il s'agit de RBAC et XACML dont l'étude est approfondie dans ce mémoire.

Les exigences du contrôle d'accès dans le domaine des dossiers médicaux électroniques ont également été visitées dans cette revue de littérature. Il en est de même des travaux précédents et similaires au nôtre. La suite du mémoire débute par le chapitre [2](#) qui présente le cas d'étude utilisé tout au long des travaux.

## CHAPITRE 1. ÉTAT DE L'ART



# Chapitre 2

## Cas d'étude

Le Centre Hospitalier Universitaire de Sherbrooke (CHUS) est un hôpital de référence pour une population de plus de 500 000 personnes. Le CHUS est connecté à d'autres centres de soins et à des services sociaux, avec lesquels il échange les données des patients. Pour assurer la sécurité des données et préserver la vie privée des patients, le CHUS a adopté une politique de contrôle d'accès qui prend en compte le consentement du patient. Dans ce chapitre, nous présentons l'architecture du système du CHUS, les exigences et les besoins du contrôle d'accès dans ce système.

### 2.1 Architecture du système

Le CHUS dispose d'une application appelée ARIANE qui gère les dossiers médicaux des patients. Cette application est connectée à une base de données qui centralise toutes les données des patients. ARIANE permet de faire la mise à jour des données des patients, de consulter les données, de transférer un patient d'un centre de soins à un autre. Le réseau de santé de la région de Sherbrooke auquel appartient le CHUS (figure 2.1), est constitué de plusieurs hôpitaux, cliniques, laboratoires privés, groupe de médecins de familles, services sociaux. Tous ces établissements sont interconnectés et chacun d'eux dispose d'une base de données contenant les dossiers médicaux de ses patients. Pour des besoins de soin, les établissements échangent entre eux les données

des patients. Ainsi un établissements A peut référer un patient à un établissement B qui dispose des moyens adéquats pour prendre en charge le patient.

La politique de sécurité définie par le CHUS permet de sécuriser les données des patients lors des échanges entre les établissements. Un seul point de contrôle d'accès défini pour tout de réseau de santé de Sherbrooke, reçoit et traite les requêtes des utilisateurs. Ce point est connecté à une base de données contenant toutes les règles qui régissent les accès aux données des patients.

Les règles régulant le contrôle d'accès dans la politique de sécurité définie par le CHUS proviennent de plusieurs sources : la loi, les établissements de soins, le patient. Tout accès au dossier d'un patient requière le consentement du patient. Les transferts de patients entre les établissements se font sur la base des accords entre les établissements. La loi intervient dans la politique en fixant des exceptions. Par exemple, la loi stipule qu'en cas d'urgence, le médecin n'a pas besoin du consentement du patient pour accéder à ses dossiers.

Au regard de la diversité des sources des règles, deux règles peuvent être en conflit lors de l'évaluation d'un même requête. Par exemple une règle de la loi autorise l'accès à une ressource et une règle du patient interdit l'accès à la même ressource. Pour permettre au système de contrôle d'accès de traiter correctement les cas de conflits, la politique de sécurité du CHUS fixe une priorité à chaque règle. En cas de conflit, c'est la règle prioritaire qui est choisie pour rendre la décision.

Les sujets dans le système du CHUS sont constitués de l'ensemble des professionnels de santé et des services affiliés, exerçant dans la région de Sherbrooke. Les sujets sont organisés sous forme hiérarchique comme l'indique la figure 2.2. Les ressources protégées dans le système sont constituées des données des patients. Les ressources sont également organisées sous forme hiérarchique, comme l'indique la figure 2.3.

## 2.2 Exigences du système

Les principales exigences de contrôle d'accès du système du CHUS sont :

**Exigence 1 :** conformité de la politique de contrôle d'accès avec la loi et les règlements.

## 2.2. EXIGENCES DU SYSTÈME

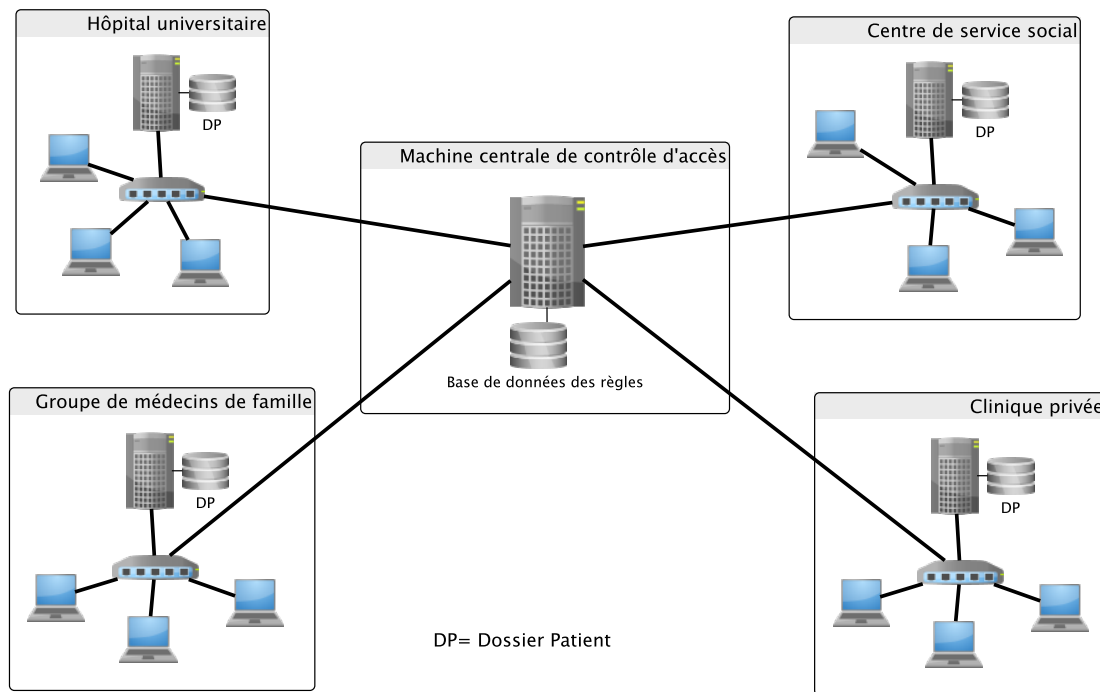


figure 2.1 – Architecture du système de contrôle d'accès du CHUS

**Exigence 2 :** la prise en compte du consentement du patient. L'accès au dossier d'un patient est conditionné par le consentement de celui-ci. Le patient peut donner accès à ses données à n'importe quel professionnel de l'hôpital.

**Exigence 3 :** le consentement du patient peut être donné par un représentant de celui-ci lorsqu'il n'est pas en mesure de le donner lui-même. Par exemple lorsque le patient est un mineur.

**Exigence 4 :** les règles de la politique de sécurité ont des priorités qui diffèrent selon qu'elles proviennent de la loi ou du patient.

**Exigence 5 :** l'accès est accordé aux utilisateurs en fonction de leur rôle, groupe de travail, association ou individuellement.

**Exigence 6 :** l'utilisation ou la communication des données sont faites conformément aux objectifs pour lesquelles ces données ont été collectées.

**Exigence 7 :** la base des données des règles contient des centaines de milliers de règles. Le système de contrôle d'accès doit permettre d'administrer facilement

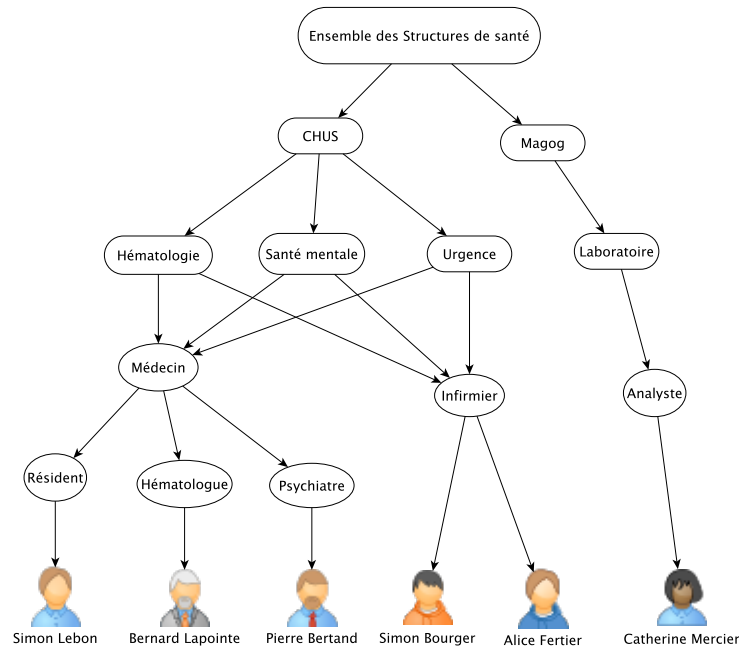


figure 2.2 – Graphe des sujets

ces règles.

**Exigence 8 :** tous les professionnels de santé de la région de sherbrooke sont des utilisateurs potentiels du système. Le moteur de contrôle d'accès reçoit à la minute, un grand nombre de demandes d'accès aux données, et il doit être à mesure de traiter efficacement l'ensemble des demandes.

**Exigence 9 :** le système doit pouvoir modéliser toutes les règles de contrôle d'accès.

**Exigence 10 :** le système doit permettre de mener facilement les audits et les investigations.

## 2.3 Besoins du système

Les besoins sont décrits à travers les scénarios ci-dessous. Dans chaque scénario, il y a le terme *Notion* qui indique ce que l'on recherche sur le modèle de contrôle d'accès à travers le scénario. Par exemple, la notion *hiérarchie des sujets* cherche à savoir comment le modèle de contrôle d'accès auquel est appliqué le cas d'étude, gère

### 2.3. BESOINS DU SYSTÈME

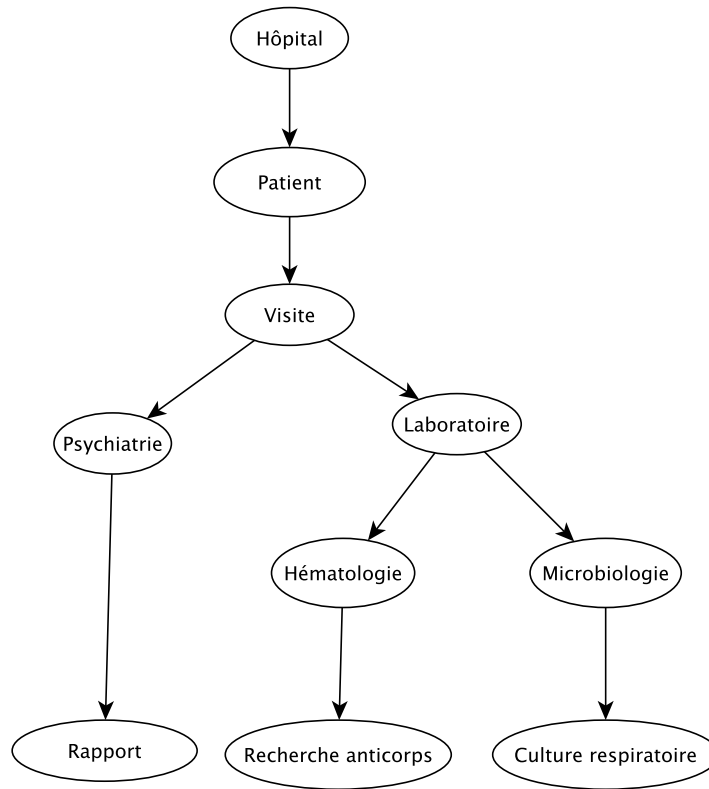


figure 2.3 – Graphe des types de ressources

la hiérarchie des sujets.

#### **Scénario 1 - Notion : traitement d'une seule règle**

Patrice est un patient qui souhaite interdire l'accès à l'ensemble de ses données personnelles (dépistage du VIH, test ADN, Examens psychiatriques) à l'ensemble des professionnels de l'hôpital.

#### **Scénario 2 - Notion : gestion de la priorité entre deux règles**

Alain est un patient qui souhaite autoriser l'accès à l'ensemble de ses données personnelles (dépistage du VIH, test ADN, Examens psychiatriques) à l'ensemble des professionnels de l'hôpital. Cela dit, il y a une loi qui précède la règle de Alain et qui interdit l'accès à ses examens psychiatriques.

### **Scénario 3 - Notion : hiérarchie des sujets**

Romain est un patient qui souhaite interdire l'accès aux « Laboratoires » à l'infirmière Alice Fertier. Cela dit, Romain avait déjà une directive qui autorisait toutes les infirmières à accéder aux Laboratoires.

### **Scénario 4 - Notion : nouvelles données**

Romain est un patient qui possède dans son dossier deux examens de laboratoire : laboratoire 1 et laboratoire 2. Il spécifie une règle autorisant les médecins à accéder à ses laboratoires. Romain ajoute ensuite à son dossier un autre laboratoire : laboratoire 3. Le Dr. Bernard Lapointe lance une requête pour accéder au laboratoire 3 qui a été ajouté au dossier après la règle définie par Romain.

### **Scénario 5 - Notion : multi-appartenance**

Simon est un patient qui a spécifié deux règles, la première interdit les professionnels travaillant à l'urgence d'accéder à son dossier et la seconde autorise les professionnels travaillant en hématologie d'accéder au dossier. Le Dr. Pierre Bertrand qui travaille à la fois en hématologie et en urgence, souhaite accéder au dossier de Simon.

### **Scénario 6 - Notion : condition**

Jeremy est un patient qui voudrait autoriser l'accès uniquement à son médecin traitant. Alors, l'accès sera automatiquement interdit dès lors que le médecin n'est plus son médecin traitant.

### **Scénario 7 - Notion : contexte**

Alice est une patiente qui voudrait autoriser l'accès aux personnels du service santé mental pour une durée de 3 jours.

Le cas d'étude est traité dans les chapitres [3](#), [4](#) et [5](#).

# Chapitre 3

## Application de RBAC au cas d'étude

Le modèle de contrôle d'accès RBAC («*Role Based Access Control*») est une norme de l'ANSI qui est largement adoptée pour contrôler les accès aux ressources dans les systèmes d'information. Il est également le modèle de contrôle d'accès qui fait l'objet de plus de recherches dans les milieux académiques. Malgré son succès, RBAC a des inconvénients et des limites quand il est utilisé pour modéliser certaines politiques de contrôle d'accès. Ce chapitre fait ressortir les forces et les faiblesses du modèle RBAC à travers son application au cas d'étude du CHUS (chapitre 2). Il comporte trois parties. La première est consacrée à la présentation de RBAC. La deuxième partie présente la modélisation en RBAC des scénarios du cas d'étude et enfin la troisième partie fait un résumé des critiques dégagées à travers cette étude.

### 3.1 Présentation de RBAC

Dans les entreprises, les agents sont regroupés par fonctions (médecins, infirmiers dans un hôpital). Chaque agent a accès aux ressources de l'entreprise au regard de la fonction qu'il occupe au sein de l'entreprise. Le contrôle d'accès RBAC reflète la structure organisationnelle de l'entreprise. L'élément clé dans RBAC est le rôle qui correspond à une fonction dans l'entreprise. Le rôle regroupe un ensemble de

### CHAPITRE 3. APPLICATION DE RBAC AU CAS D'ÉTUDE

permissions ou droits d'accès et les attribue aux agents appelés utilisateurs dans RBAC en tenant compte de leur fonction dans l'entreprise. Un utilisateur assigné à un rôle, bénéficie des permissions affectées à ce rôle, en activant le rôle dans une session. Quand un utilisateur veut accéder à un objet dans un système contrôlé par RBAC, il envoie une requête comprenant son identité, le nom de l'objet et l'action qu'il veut exécuter. Par exemple  $\langle Bob, lire, fichier1.txt \rangle$  où *Bob* représente l'identité de l'utilisateur, *lire* représente l'action que l'utilisateur souhaite exécutée et *fichier1.txt* représente l'objet auquel l'utilisateur veut accéder. La figure 3.1 représente le flux d'information dans un système RBAC. La description de la figure se présente comme suit :

1. *Bob* se connecte à l'application qui protège les objets et envoie sa requête. En se connectant, *Bob* ouvre une session et active un certain nombre de rôles qui lui sont affectés.
2. L'application qui protège les objets envoie à la machine RBAC, la requête de *Bob* et les rôles activés par *Bob* dans sa session. La machine évalue la requête en vérifiant s'il existe dans sa base de données des règles, une permission (*lire fichier1.txt*) qui est assignée à un des rôles activés par *Bob*. En cas de réponse positive à cette vérification, la machine RBAC décide d'autoriser la lecture du fichier demandé ; sinon elle l'interdit.
3. La décision prise par la machine RBAC est envoyée à l'application qui protège l'objet. Si la décision interdit l'accès à l'objet, une réponse d'interdiction est envoyée à l'utilisateur et le processus s'arrête ici.
4. Si la décision de l'évaluation autorise l'accès à l'objet, l'application qui protège l'objet requière l'objet auprès de l'entité qui contient les objets.
5. L'objet demandé est retourné à l'application qui protège les objets.
6. L'objet est retourné à l'utilisateur.

Le modèle RBAC comprend quatre composantes. Le **core RBAC** est la composante de base qu'on retrouve dans tout système RBAC. La **hiérarchie RBAC** permet de modéliser l'héritage entre les rôles. Les composantes **séparation statique des tâches** (SSD : «*static separation of duties*» ) et **séparation dynamique des tâches** (DSD : «*dynamic separation of duties*»), permettent de résoudre les problèmes de conflits entre les rôles.



### 3.1. PRÉSENTATION DE RBAC

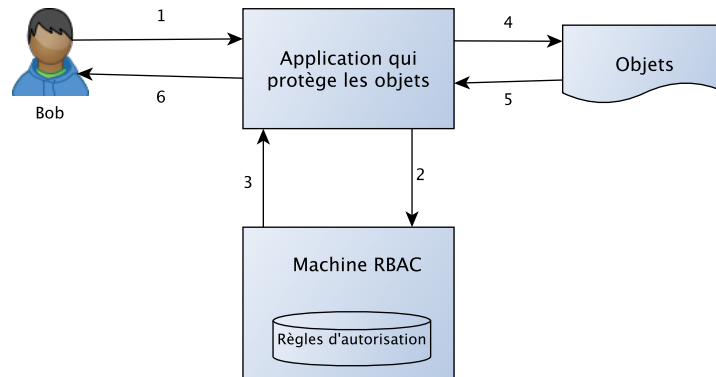


figure 3.1 – Flux d’information dans un système RBAC

- Le **core RBAC** ou le **noyau de RBAC** définit une collection minimale d’éléments, d’ensembles éléments et de relations nécessaires à la mise en place d’un système RBAC : *utilisateurs*, *rôles*, *objets*, *opérations* et *permissions*. La figure 3.2 présente le modèle entités-associations du core RBAC. Le core RBAC introduit aussi un ensemble de *sessions* où chaque session est une association entre un utilisateur et un sous-ensemble de rôles qui lui sont attribués.

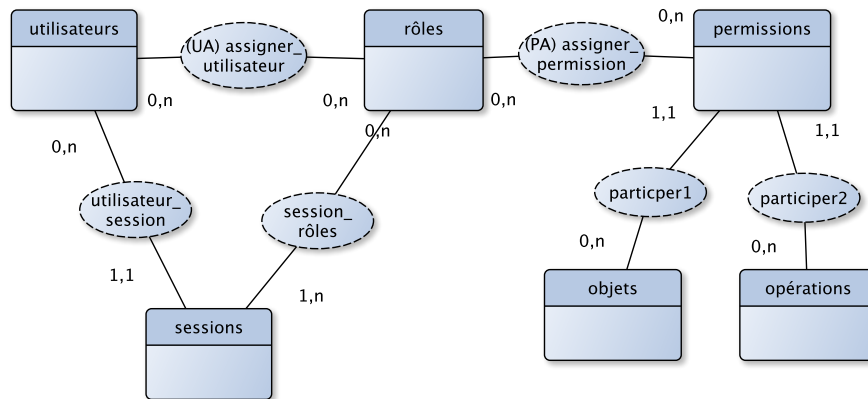


figure 3.2 – Modèle entités-associations du core RBAC

- La **hiérarchie RBAC** permet d’introduire des relations d’héritage entre les rôles, dans le but de faciliter l’administration du système. Les relations d’héritage entre les rôles sont établies de façon analogue à l’organisation des métiers

### CHAPITRE 3. APPLICATION DE RBAC AU CAS D'ÉTUDE

en entreprise, où l'on trouve une hiérarchie organisée suivant le degré d'autorité et de responsabilité. Un rôle A hérite d'un rôle B signifie que toutes les permissions de B sont aussi des permissions de A et tous les utilisateurs pouvant jouer le rôle A peuvent aussi jouer le rôle B. Dans la figure 3.3, le rôle *médecin cardiologue* hérite du rôle *médecin*. Anne qui est médecin bénéficie seulement des permissions 1 et 2, alors que Bob qui est médecin cardiologue bénéficie des permissions 1, 2 et 3.

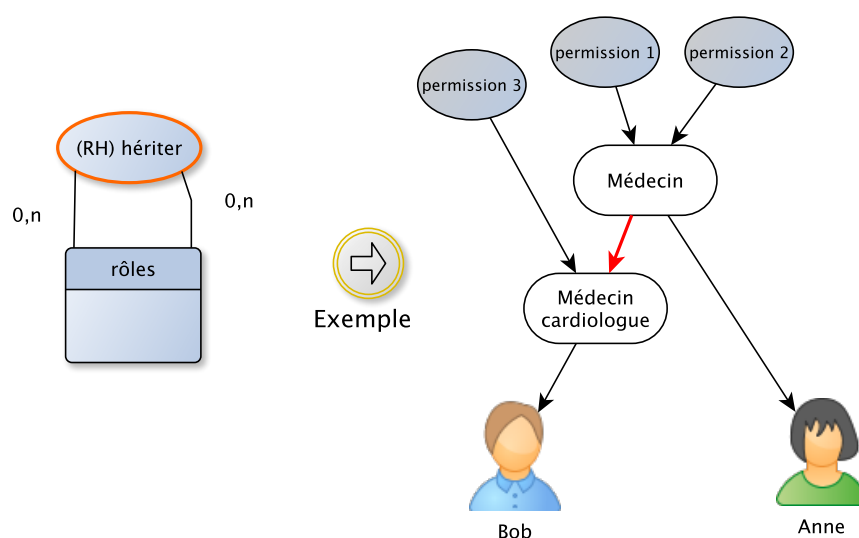


figure 3.3 – Héritage entre les rôles dans hiérarchie RBAC

Il existe deux types de hiérarchie de rôles : la hiérarchie générale et la hiérarchie limitée. La hiérarchie générale introduit la notion d'héritage multiple. Ce qui veut dire qu'un rôle  $R_1$  peut hériter en même temps de  $R_2$ , de  $R_3$ , etc. La relation d'héritage est un ordre partiel, ce qui veut dire que si  $R_1$  hérite de  $R_2$  et  $R_2$  hérite de  $R_3$  alors par fermeture transitive de l'ordre partiel,  $R_1$  hérite de  $R_3$ . La hiérarchie limitée impose une contrainte à la hiérarchie générale. Avec la hiérarchie limitée, un rôle  $R_1$  hérite directement d'un seul rôle  $R_2$ . Si  $R_1$  hérite de  $R_2$  et  $R_1$  hérite aussi de  $R_3$  alors  $R_2=R_3$ .

- **La séparation statique des tâches (SSD)** définit des contraintes dans l'assignation des rôles aux utilisateurs. Elle est utilisée dans les cas où il y a

### 3.1. PRÉSENTATION DE RBAC

des rôles mutuellement exclusifs et qui ne peuvent pas être assignés à un même utilisateur. Par exemple, il est interdit à une même personne d'exercer en même temps le rôle de chirurgien et le rôle d'anesthésiste lors d'une opération. La figure 3.4 présente l'ajout de la SSD au core RBAC. La SSD intervient lors de l'assignation des rôles à utilisateurs pour empêcher que deux rôles conflictuels ne soient affectés à un même utilisateur et également lors de l'héritage entre les rôles pour empêcher de faire hériter deux rôles conflictuels.

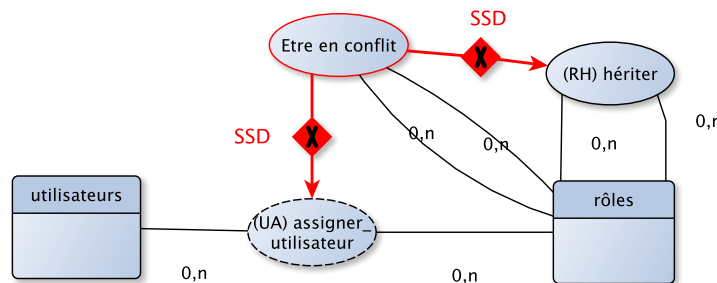


figure 3.4 – Modélisation de la séparation statique des tâches en RBAC

- La **séparation dynamique de tâches (DSD)** impose une contrainte lors de l'activation dans une session, des rôles que possède un utilisateur. Un même utilisateur peut se voir assigner deux rôles mutuellement exclusifs mais avec la DSD l'utilisateur ne pourra pas activer les deux rôles dans une même session et par conséquent, ne pourra pas exercer les deux rôles en même temps. La figure 3.5 présente l'ajout de la DSD au core RBAC.

Comparaison faite avec les modèles DAC et MAC, l'avènement de RBAC est une grande révolution dans le domaine du contrôle d'accès. Le regroupement de plusieurs utilisateurs en un seul rôle facilite la gestion des utilisateurs dans les grands systèmes. Lorsqu'un utilisateur change de fonction, la procédure de mise à jour est simple ; il suffit de l'enlever de son ancien rôle et de l'assigner au rôle dédié à sa nouvelle fonction. L'utilisation des rôles permet aussi de vérifier facilement qui a accès à quoi dans le système. Avec RBAC, il est facile d'assurer le principe de moindre privilèges qui veut que chaque utilisateur ait le minimum de privilèges lui permettant de remplir sa fonction au sein de l'entreprise. Le modèle RBAC a connu un grand succès, tant dans

## CHAPITRE 3. APPLICATION DE RBAC AU CAS D'ÉTUDE

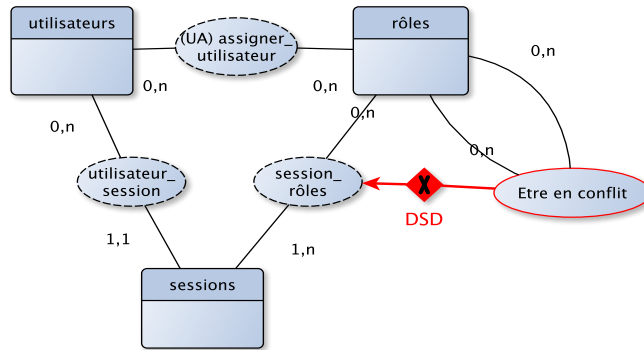


figure 3.5 – Modélisation de la séparation dynamique des tâches en RBAC

l'industrie que dans les recherches académiques [43]. Dans le domaine de la santé, c'est le modèle le plus largement utilisé [6, 7]. La section suivante présente un bref aperçu de l'utilisation de RBAC dans le domaine de la santé.

### 3.2 Application de RBAC au cas d'étude

L'application de RBAC au cas d'étude du CHUS présenté au chapitre 2 consiste à modéliser en RBAC les scénarios de ce cas d'étude. Les scénarios sont décrits en utilisant des schémas. Le tableau 3.1 décrit les principales figures utilisées dans les schémas.


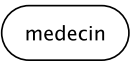

	désigne une permission
	désigne un rôle
	désigne un utilisateur

tableau 3.1 – Description des figures utilisées pour représenter les scénarios

## 3.2. APPLICATION DE RBAC AU CAS D'ÉTUDE

### 3.2.1 Scénario 1

Notion : Traitement d'une seule règle

Les exigences du scénario 1 :

$E_{1.1}$  : *Patrice est un patient qui souhaite interdire l'accès à l'ensemble de ses données personnelles (dépistage du VIH, test ADN, examens psychiatriques) à l'ensemble des professionnels de l'hôpital.*

#### Solution du scénario 1

**Etat du système avant application de  $E_{1.1}$  :** tout le personnel de l'hôpital a accès à l'ensemble des données personnelles de Patrice (dépistage du VIH, test ADN, examens psychiatriques). Dans le système RBAC, toutes les permissions d'accès aux données de Patrice sont assignées au rôle *ToutProfessionnel*. Ce rôle contient toutes les permissions qui sont affectées à l'ensemble des professionnels de l'hôpital. La modélisation est représentée par la figure 3.6.  $P_i$  désigne la permission d'accès à la donnée  $i$  (exemples *dépistage VIH 1*, *dépistage VIH 2*, *test ADN 1*).

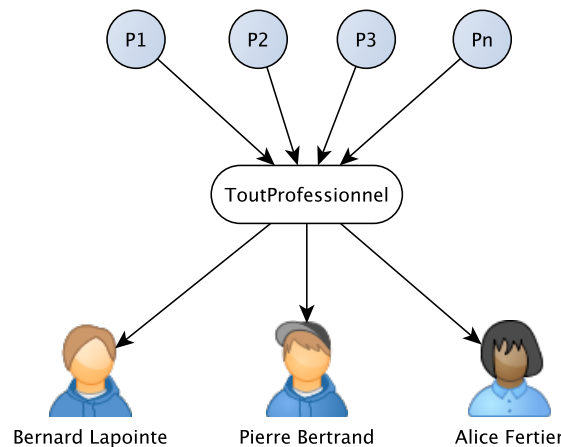


figure 3.6 – Accès accordé à tout le personnel de l'hôpital sur les données de Patrice

**Application de  $E_{1.1}$  :** suppression des autorisations préalablement accordées aux professionnels de l'hôpital. L'application de  $E_{1.1}$  est illustrée par la figure 3.7

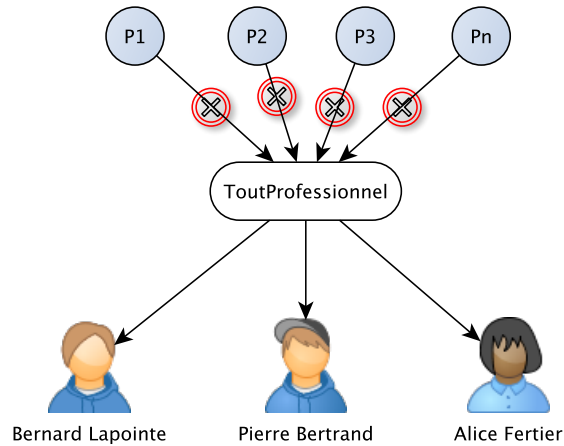


figure 3.7 – Retrait des autorisations d'accès aux données de Patrice

**Etat du système après l'application de  $E_{1.1}$  :** il n'y a aucune autorisation d'accès aux données de Patrice dans le système RBAC. L'exigence  $E_{1.1}$  a été mise en œuvre dans le système mais il n'y a aucune trace dans le système qui montre l'existence de  $E_{1.1}$ . Il y a par conséquent un risque d'une perte possible de  $E_{1.1}$  par oubli. Cette perte possible s'explique par le fait que les interdictions ne sont pas modélisées en RBAC. Le fait de perdre  $E_{1.1}$  peut conduire à sa violation.  $E_{1.1}$  peut être une règle d'une forte priorité et comme il est possible de l'oublier, l'administrateur des règles peut représenter une permission moins prioritaire que  $E_{1.1}$  et l'usage de cette permission violera  $E_{1.1}$ .

## 3.2.2 Scénario 2

Notion : Gestion de la priorité entre deux règles.

Les exigences du scénario 2 :

$E_{2.1}$  : *La loi interdit l'accès aux examens psychiatriques de Alain.*

$E_{2.2}$  : *Alain souhaite autoriser l'accès à l'ensemble de ses données personnelles à l'ensemble des professionnels de l'hôpital.*

$E_{2.3}$  : *La loi est prioritaire par rapport aux règles du patient*

## Solution du scénario 2

### 3.2. APPLICATION DE RBAC AU CAS D'ÉTUDE

**Etat du système avant :** il n'y a aucune permission sur les données de Alain qui existe dans le système.

#### **Application des exigences du scénario 2 :**

- Traitement de l'exigence  $E_{2,1}$  : il consiste à une suppression des permissions d'accès aux examens psychiatriques d'Alain. Mais comme il n'y a aucune permission sur ces données, il n'y a pas d'opération à effectuer
- Traitement de l'exigence  $E_{2,2}$  : il consiste à créer les autorisations d'accès sur les données de Alain.  $P_1, P_2$  et  $P_3$  désignent respectivement les permissions d'accès au *dépistage du VIH 1*, au *test ADN 1* et à l'*examen psychiatrique 1* de Alain. Si l'administrateur du système se rappelle de  $E_{2,1}$  (cette exigence existe bien avant  $E_{2,2}$ ), il ne va pas créer  $P_3$ . Mais comme il n'existe aucune représentation dans le système qui marque l'existence de  $E_{2,1}$ , il y a une forte probabilité que l'administrateur l'oublie et crée  $P_3$ . La figure 3.6 illustre également la représentation de  $E_{2,2}$ .
- Traitement de l'exigence  $E_{2,3}$  : cette exigence ne pourra pas être représentée car il n'existe pas de priorité entre les permissions en RBAC.

**Etat du système après l'application du scénario 2 :** tout le personnel a accès aux examens psychiatriques de Alain. Il y a une violation de la loi qui pourtant est prioritaire par rapport à la règle de Alain. Cette violation est liée à l'impossibilité de modéliser les interdictions et les priorités en RBAC. Lorsque les règles proviennent de plusieurs sources (loi, hôpital, patient), il peut arriver que certaines règles soient en conflit entre elles. On a besoin que le système gère les conflits entre les règles en utilisant la priorité assignée à chaque règle. Le fait de ne pas pouvoir modéliser les priorités entre règles peut être source de violation de la politique de sécurité.

#### 3.2.3 Scénario 3

Notion : Hiérarchie des sujets.

Les exigences du scénario 3 :

$E_{3,1}$  : *Romain est un patient qui souhaite interdire l'accès aux « Laboratoires » à l'infirmière Alice Fertier.*

$E_{3.2}$  : Romain avait déjà une directive qui autorisait toutes les infirmières à accéder aux Laboratoires

### Solution du scénario 3

**Etat du système avant :** il n'y a aucune permission dans le système sur les laboratoires de Romain.

**Application des exigences du scénario 3 :**  $E_{3.2}$  est traité en premier lieu car comme dit dans le scénario,  $E_{3.2}$  précède  $E_{3.1}$ .

- Traitement de l'exigence  $E_{3.2}$  : les permissions d'accès aux laboratoires ( $P_5$ ,  $P_6$ ) de Romain sont directement assignées au rôle *infirmière* comme le montre la figure 3.8.

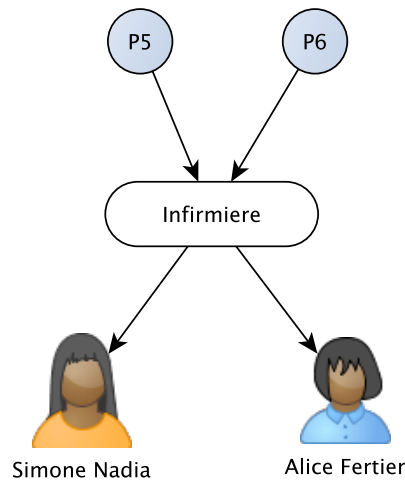


figure 3.8 – Autorisation d'accès aux laboratoires de Romain

- Traitement de l'exigence  $E_{3.1}$  : il consiste à :
  1. Suppression des permissions  $P_5$  et  $P_6$  du rôle *infirmière*.
  2. Création d'un nouveau rôle  $R005$  correspondant aux infirmières qui ont accès aux laboratoires de Romain.
  3. Affectation de  $P_5$  et  $P_6$  au rôle  $R005$ .



### 3.2. APPLICATION DE RBAC AU CAS D'ÉTUDE

4. Établir une relation d'héritage entre  $R005$  et *infirmière*.  $R005$  hérite de *infirmière*

La figure 3.9 illustre la modélisation de  $E_{3.1}$ .

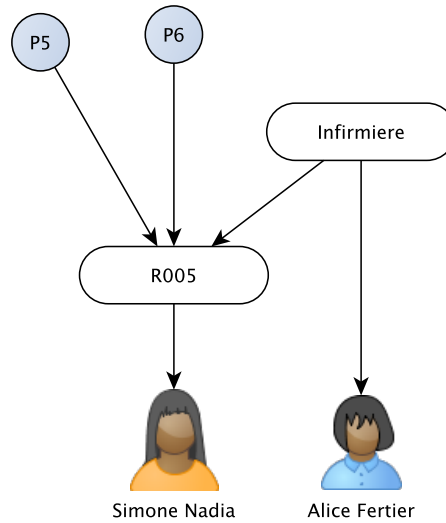


figure 3.9 – Restriction des autorisations d'accès aux laboratoires de Romain à un groupe d'infirmières

**Etat du système après l'application du scénario 3 :** toutes les infirmières ont accès aux laboratoires de Romain, sauf Alice fertier. La mise en oeuvre de  $E_{3.1}$  requière l'accomplissement de plusieurs actions : vérifications afin d'isoler Alice fertier de tous les rôles qui héritent des permissions d'accès aux laboratoires de Romain, création de nouveau rôle, réaffectations des permissions. Ces actions rendent l'administration du système difficile dans les systèmes larges comme ceux de la santé qui peuvent contenir plusieurs milliers de règles. Certains systèmes comme celui du réseau de santé de Walloom [46] délègue entièrement l'administration des règles aux patients. La mise en oeuvre du scénario 3 par un patient sera difficile à réaliser et possiblement source d'erreurs qui vont entraîner la violation des données. Les patients n'ont pas assez de connaissances techniques pour effectuer toutes les actions ayant conduit à la mise en oeuvre du scénario 3.

### 3.2.4 Scénario 4

Notion : nouvelles données

Les exigences du scénario 4 :

$E_{4.1}$  : *Romain autorise les médecins à accéder à ses laboratoires.*

#### Solution du scénario 4

**Etat du système avant :** il n'y a aucune permission dans le système sur les laboratoires de Romain.

**Application des exigences du scénario 4 :** création des permissions sur les laboratoires de Romain. A la formulation de  $E_{4.1}$ , il n'y avait que deux laboratoires dans le dossier de Romain.  $P_1$  et  $P_2$  représentent les deux permissions créées pour les deux laboratoires de Romain. L'ajout du troisième laboratoire de Romain nécessite la création d'une nouvelle permission pour cette nouvelle donnée. Soit  $P_3$  la permission d'accès au troisième laboratoire de Romain. Affecter ensuite  $P_3$  au rôle *médecin*. L'administrateur des règles de contrôle d'accès peut ne pas être informé de l'ajout de la nouvelle donnée et par conséquent, il ne créera pas  $P_3$ . Sans  $P_3$  le nouveau laboratoire ne sera pas accessible, car tout est interdit par défaut dans RBAC. La modélisation du scénario 4 est représentée par le figure [3.10](#).

**Etat du système après l'application du scénario 4 :** si l'administrateur des règles a créé  $P_3$ , alors le médecin Bernard Lapointe aura accès au nouveau laboratoire de Romain. Sinon l'accès lui sera interdit. Le fait que RBAC ne modélise pas la hiérarchie des ressources, entraîne une forte probabilité que les exigences préalablement définies ne soient pas appliquées aux nouvelles données.

### 3.2.5 Scénario 5

Notion : Multi-appartenance

Les exigences du scénario 5 :

### 3.2. APPLICATION DE RBAC AU CAS D'ÉTUDE

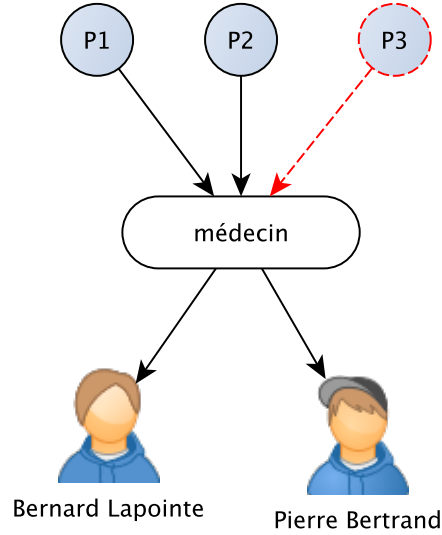


figure 3.10 – Modélisation du scénario 4

$E_{5.1}$  : Simon interdit d'abord les professionnels travaillant à l'urgence d'accéder à son dossier.

$E_{5.2}$  : Simon autorise ensuite les professionnels travaillant en hématologie d'accéder au dossier

#### Solution du scénario 5

**Etat du système avant :** c'est l'état à l'issue de la modélisation de  $E_{5.1}$ . La figure 3.11 représente la modélisation de  $E_{5.1}$ . Le rôle *urgence* est dédié à l'ensemble des professionnels travaillant à l'urgence. Soit  $\mathcal{P}$  l'ensemble des permissions d'accès au dossier de Simon. La modélisation de  $E_{5.1}$  consiste à supprimer du rôle *urgence* toutes les permissions  $pi$  tel que  $pi \in \mathcal{P}$ .

**Application de  $E_{5.2}$  :** l'application de  $E_{5.2}$  en RBAC consiste à assigner toutes les permissions d'accès au dossier de Simon au rôle *hématologie* auquel sont affectés les professionnels travaillant en Hématologie. La figure 3.12 représente la modélisation de  $E_{5.1}$  et  $E_{5.2}$ .

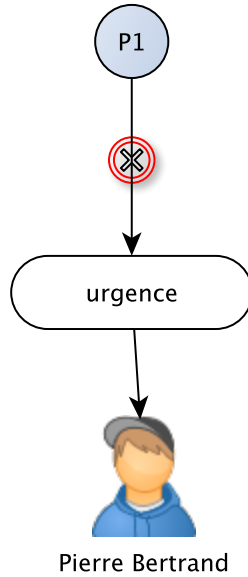


figure 3.11 – Modélisation de  $E_{5.1}$

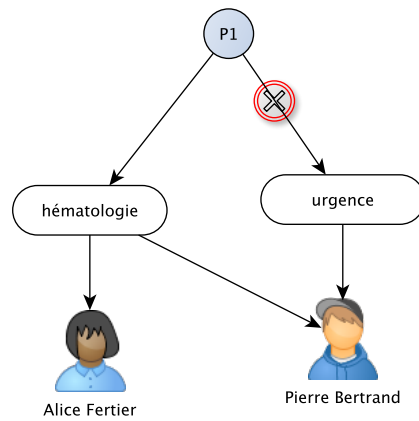


figure 3.12 – Modélisation de  $E_{5.1}$  et  $E_{5.2}$

**Etat du système après la mise en œuvre du scénario 5 :** le médecin Pierre Bertrand qui travaille dans les deux services (Hématologie et Urgence) est autorisé à accéder au dossier de Simon en violation de  $E_{5.1}$ . L'autorisation accordée à Pierre Bertrand est un choix arbitraire du système RBAC, car il est interdit par  $E_{5.1}$  et

### 3.2. APPLICATION DE RBAC AU CAS D'ÉTUDE

autorisé par  $E_{5.2}$ . La décision du système se justifie par le fait que  $E_{5.1}$  n'est pas représentée. L'idéal serait que le système représente les deux réglés et que pour gérer le conflit, le système invite le patient à se prononcer sur la primauté d'une exigence sur l'autre.

#### 3.2.6 Scénario 6

Notion : Condition

Les exigences du scénario 6 :

$E_{6.1}$  : *Jeremy est un patient qui souhaite autoriser l'accès juste à son médecin traitant. Alors, l'accès sera automatiquement interdit une fois le médecin n'est plus son médecin traitant.*

#### Solution du scénario 6

**Etat du système avant :** nous supposons qu'aucune permission d'accès aux données de Jeremy n'est encore créée. Dans les cas où ces permissions existent déjà, elles doivent être retirées de l'ensemble des rôles.

**Application de  $E_{6.1}$  :** l'application de  $E_{6.1}$  consiste à créer un rôle *médecinJeremy* et à assigner à ce rôle toutes les permissions d'accès aux données de Jeremy. La figure 3.13 représente la modélisation de  $E_{6.1}$ .

**Etat du système après modélisation du scénario 6 :** la représentation a permis d'autoriser un médecin à accéder au dossier de Jeremy. Cependant, lorsque Jeremy va changer de médecin traitant, il n'y aura pas de mise à jour automatique. L'administrateur du système doit le faire manuellement. La mise à jour manuelle est non seulement difficile dans les systèmes comportant plusieurs patients, mais aussi elle peut être source de violation de la règle du patient. En effet, si Jeremy change de médecin traitant et que l'administrateur des règles de contrôle d'accès ne fait pas cette mise à jour dans le système RBAC, l'ancien médecin continuera à accéder aux données de Jeremy.

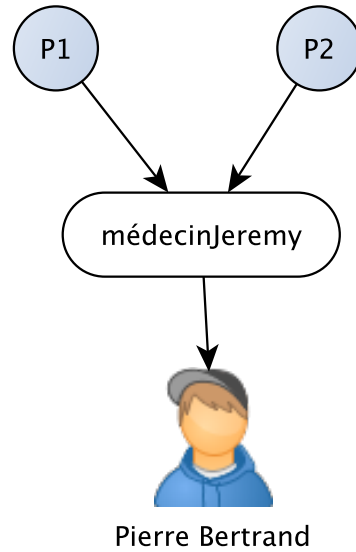


figure 3.13 – Modélisation du scénario 6

### 3.2.7 Scénario 7

Notion : Contexte temps

Les exigences du scénario 7 :

$E_{7.1}$  : *Alice est une patiente qui voudrait autoriser l'accès aux personnels du service de santé mentale pour une durée de 3 jours.*

#### Solution du scénario 7

**Etat du système avant :** nous supposons qu'aucune permission n'est encore créée sur les données de Alice.

**Application de  $E_{7.1}$  :** l'application de  $E_{7.1}$  consiste à créer les permissions d'accès aux données d'Alice et à les assigner au rôle dédié aux personnels du service de santé mentale. La norme RBAC n'introduit pas la notion de temps dans la modélisation des permissions. Les permissions créées doivent être supprimées après trois jours.

### 3.3. CRITIQUES RBAC

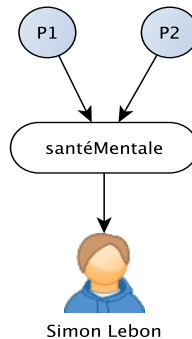


figure 3.14 – Modélisation du scénario 7 sans le contexte temps

**Etat du système après modélisation du scénario 7 :** le personnel du service de santé mentale a accès aux données de Alice. Si à l’issue du troisième jour l’administrateur oublie de modifier les permissions, le personnel du service de santé mentale va continuer à accéder aux données de Alice en violation de sa règle. Il serait difficile voire impossible dans les systèmes comportant plusieurs règles, d’administrer des permissions qui changent avec le temps.

## 3.3 Critiques RBAC

Ce cas d’étude révèle des limites de RBAC qui se résument à : impossibilité de modéliser les règles d’interdiction, impossibilité de modéliser la hiérarchie de ressources, impossibilité de modéliser des règles dépendantes d’un contexte.

**Impossibilité de modéliser les règles d’interdiction :** l’interdiction est la règle par défaut en RBAC. Tout ce qui n’est pas explicitement permis est interdit et tout ce qui est interdit n’est pas représenté dans le système. L’impossibilité de représenter les règles d’interdiction entraîne les conséquences suivantes :

1. La perte des règles d’interdiction : le système RBAC ne contenant pas les traces des règles d’interdiction, il y a une forte probabilité que ces règles soient oubliées au fil du temps. La perte des règles d’interdiction ouvre la voie à toutes sortes de violations du contrôle d’accès.

### CHAPITRE 3. APPLICATION DE RBAC AU CAS D'ÉTUDE

2. La violation des règles d'interdiction : lorsque deux règles  $R_1$  et  $R_2$  portent sur un même objet  $O$ ,  $R_1$  interdit l'accès à  $O$  et  $R_2$  autorise l'accès à  $O$ , alors  $R_2$  a préséance sur  $R_1$  à cause du fait que  $R_1$  n'est pas modélisée. Prenons l'exemple du scénario 2. De part le fait que la loi n'est pas représentée, il est facile de l'oublier et, en représentant la règle de Alain, tous les infirmiers ont accès aux examens psychiatriques de Alain en violation de la loi.

**L'explosion des rôles :** en RBAC, il n'y a pas de possibilité d'assigner les permissions directement à un utilisateur. Lorsqu'une règle porte sur un seul utilisateur, il faut créer un rôle pour cet utilisateur. Il en est de même lorsqu'un patient autorise un sous-groupe de médecins à accéder à son dossier ; il faut créer un nouveau rôle pour ce sous-groupe de médecins et appliquer une contrainte de séparation des tâches afin d'éviter que tous les médecins n'accèdent au dossier du patient. La création de nouveaux rôles entraîne l'explosion des rôles qui peut rendre l'administration du système difficile.

**Impossibilité de modéliser les priorités entre les règles :** la priorité sert à la gestion des conflits entre règles. Il y a conflit lorsque pendant l'évaluation d'une requête, le système se retrouve avec deux ou plusieurs règles applicables ayant des effets différents. Certaines règles autorisent l'accès pendant que d'autres l'interdisent. L'utilisation de la priorité permet au système de déterminer la règle devant être utilisée pour rendre la décision. Le conflit tel que défini n'existe pas en RBAC, car il ne modélise pas les interdictions. Cependant, il y a d'autres formes de conflits en RBAC présentés par Jiong et Chen-hual [34]. Les conflits en RBAC sont liés à la définition des contraintes. Par exemple, si nous avons deux contraintes  $C_1$  et  $C_2$  définies de la sorte :

- $C_1$  : les rôles  $R_1$  et  $R_2$  sont contradictoires en raison de la séparation des tâches, c'est à dire que les deux rôles ne peuvent pas être assignés à un même utilisateur ;
- $C_2$  :  $R_1$  est un préalable à  $R_2$  c'est à dire que  $R_2$  peut être assigné à un utilisateur seulement si  $R_1$  a été assigné à cet utilisateur.

$C_1$  et  $C_2$  sont incompatibles. RBAC ne propose pas un mécanisme de résolution de



### 3.3. CRITIQUES RBAC

ces conflits.

**Impossibilité de modéliser la hiérarchie de ressources :** lorsque les données sont organisées sous forme hiérarchique, en définissant une règle sur un nœud parent, cette règle s’applique sur les nœuds fils. Ce qui n’est pas le cas en RBAC. La même règle qui a été définie sur le nœud parent doit être définie sur chacun des nœuds fils. Cette limite de RBAC provoque :

1. la multiplication du nombre de règles à créer pouvant entraîner des difficultés dans l’administration
2. la non prise en compte des nouvelles données dans les permissions : considérant l’exemple du scénario 3, en cas d’ajout d’un nouveau laboratoire de Romain, l’administrateur doit se rappeler la règle de Romain et créer une permission pour le nouveau laboratoire, sinon le nouveau laboratoire ne sera pas concerné par la règle de Romain et toute demande d’accès au nouveau laboratoire sera interdite.

**Impossibilité de modéliser les règles dépendant d’un contexte :** les règles dépendant d’un contexte changent suivant la valeur prise par le contexte à un moment donné. C’est le cas des règles qui dépendent du temps ou d’un lieu. Les permissions en RBAC sont statiques. Pour que l’état du système change en fonction de la valeur d’un événement contextuel, il faut l’intervention manuelle de la part de l’administrateur des règles. Par exemple, en accordant des autorisations pour 3 jours, à la date d’expiration des 3 jours, il faut retirer manuellement ces autorisations.

Au regard des limites de RBAC, NIST a appelé à l’élaboration d’un modèle de contrôle d’accès qui prend en compte l’utilisation d’attributs et qui maintient les avantages de RBAC [37]. L’appel de NIST date de juin 2010 et nous n’avons pas encore connaissance d’une solution en réponse à cet appel. Mais bien avant cet appel, il existe des modèles comme ABAC [33] et XACML [50], qui décrivent les politiques de contrôle d’accès en se basant sur les attributs (ressource, sujet, environnement). XACML est étudié au chapitre 4.

## CHAPITRE 3. APPLICATION DE RBAC AU CAS D'ÉTUDE

# Chapitre 4

## Application de XACML au cas d'étude

XACML («*eXtensible Access Control Markup Language*») est un langage basé sur XML et dédié au Contrôle d'Accès (CA). Il permet de décrire les politiques du CA, les requêtes sur les ressources et les réponses aux requêtes. En plus du langage, XACML offre une architecture pour guider la mise en place d'un mécanisme de CA. XACML est l'un des langages de contrôle d'accès les plus populaires. Il fait l'objet de beaucoup de recherches dans le domaine académique. XACML a été standardisé par OASIS («*Organization for the Advancement of Structured Information Standards*»). C'est un langage très riche qui permet de modéliser des règles comportant des conditions complexes ; et par conséquent, de résoudre des problèmes complexes de contrôle d'accès dans l'entreprise. Cependant, beaucoup d'entreprises hésitent à adopter la solution XACML à cause de la complexité du langage. Parmi les entreprises qui utilisent XACML, se trouve AXIOMATIC [9] qui se base essentiellement sur ce modèle pour développer des solutions au profit de ses clients. La compagnie Boeing a adopté XACML comme architecture de base de leur système de sécurité [56]. Oracle et NextLabs [26] utilisent également la solution XACML. Ce chapitre comprend la présentation de XACML, la description de la mise en oeuvre de XACML, la modélisation en XACML du cas d'étude et les critiques de XACML.

## 4.1 Présentation de XACML

XACML est très complexe. Une brève introduction de XACML disponible sur le site OASIS, explique succinctement XACML. Les travaux de Abd El-Aziz et Kannan[16] expliquent également XACML. Les sous-sections qui suivent présentent le langage XACML (sous-section 4.1.1), l'architecture du modèle de contrôle d'accès XACML (sous-section 4.1.2), les profils XACML (sous-section 4.1.3), le processus de prise de décision dans XACML (sous-section 4.1.4).

### 4.1.1 Langage XACML

XACML définit un langage de politiques qui permet de décrire les règles de la politique du contrôle d'accès, et un langage pour les requêtes et les réponses, qui décrit le format d'une requête XACML et le format d'une réponse XACML.

Le langage de politiques XACML se compose de : *ensemble de politiques* («*policy set*»), *politique* («*policy*»), *règle* («*rule*»), *cible* («*target*»), *effet* («*effect*»), *condition*, *obligation*, *avis* («*advice*»), *algorithme de combinaison* («*combining algorithm*»). Les relations entre les composants du langage de politiques XACML sont présentées dans le diagramme UML de la figure 4.1. L'exemple suivant est utilisé pour décrire comment les politiques de contrôle d'accès sont représentées en XACML.

**Exemple 1 :** *Romain est un patient qui souhaite interdire l'accès à ses résultats de laboratoires à l'infirmière Alice Fertier. Cela dit, Romain avait déjà une directive qui autorisait toutes les infirmières à accéder à ses résultats de laboratoires.*

Dans l'exemple 1, il y a deux exigences qui sont :

$E_1$  : *Romain est un patient qui souhaite interdire l'accès à ses résultats de laboratoires à l'infirmière Alice Fertier.*

$E_2$  : *Romain avait déjà une directive qui autorisait toutes les infirmières à accéder à ses Laboratoires.*

L'exemple 1 est modélisé dans le programme 4.1.

Les sous-sections qui suivent décrivent les principaux composants du langage XACML.

## 4.1. PRÉSENTATION DE XACML

programme 4.1 – Modélisation de l'exemple1 en XACML

```
0 <PolicySet PolicySetId="1" PolicyCombiningAlgId="first-applicable">
1   <Target />
2   <Policy PolicyId="exemple1" RuleCombiningAlgId="deny-overrides">
3     <Target>
4       <AnyOf>
5         <AllOf>
6           <Match MatchId="function:string-regex-match">
7             <AttributeValue> Rom_001/Labo/* </AttributeValue>
8             <AttributeDesignator MustBePresent="false"
9               Category="attribute-category:resource"
10              AttributeId="resource:resource-id">
11           </Match>
12         </AllOf>
13       </AnyOf>
14     </Target>
15     <Rule RuleId="E1" Effect="Deny">
16       <Target />
17       <Condition>
18         <Apply FunctionId="function:any-of">
19           <Function FunctionId="function:string-equal"/>
20           <AttributeValue> Alice Fertier </AttributeValue>
21           <AttributeDesignator
22             Category="subject-category:access-subject"
23             AttributeId="subject:subject-id">
24         </Apply>
25       </Condition>
26     </Rule>
27     <Rule RuleId="E2" Effect="Permit">
28       <Target />
29       <Condition>
30         <Apply FunctionId="function:any-of">
31           <Function FunctionId="function:string-equal"/>
32           <AttributeValue> infirmiere </AttributeValue>
33           <AttributeDesignator
34             Category="subject-category:access-subject"
35             AttributeId="subject:role">
36         </Apply>
37       </Condition>
38     </Rule>
39   </Policy>
40 </PolicySet>
```

### La règle

Elle correspond à une exigence dans une politique de contrôle d'accès. Chacune des exigences  $E_1$  et  $E_2$  correspond à une règle. La plage de la ligne 15 à la ligne 26

## CHAPITRE 4. APPLICATION DE XACML AU CAS D'ÉTUDE

représente  $E_1$ . La plage de la ligne 27 à la ligne 38 représente  $E_2$ . Une règle XACML comprend :

Une cible : elle définit l'ensemble des requêtes auxquelles s'applique la règle. L'ensemble des requêtes auxquelles s'applique la règle est constitué des requêtes dont les valeurs des attributs *ressource*, *sujet*, *action* s'apparient («*match*» en anglais) avec les valeurs de ces mêmes attributs présents dans la *cible* de la règle. Par définition, l'attribut *sujet* d'une requête s'apparie avec l'attribut *sujet* d'une règle, si la fonction booléenne utilisée dans le «*match*» de la cible retourne *VRAI*. La *cible* d'une règle peut ne pas être définie, dans ce cas la *cible* de la règle est la même que la *cible* de la politique. Dans la représentation de l'exemple 1, les cibles de  $E_1$  et  $E_2$  ne sont pas définies. Ces deux règles sont contenues dans la politique identifiée par exemple 1. La cible de exemple 1 est définie dans la plage de la ligne 3 à la ligne 14. Pour que la politique de l'exemple 1 soit considérée dans l'évaluation d'une requête, il faut que la ressource demandée dans la requête, comparée à la valeur *Rom\_001/Labo/\** (ligne 7) en utilisant la fonction booléenne **string-regex-match**, retourne *VRAI*. *Rom\_001/Labo/\** représente l'ensemble des laboratoires de Romain. En utilisant la fonction **string-regex-match**, toute requête dont la valeur de la ressource contient *Rom\_001/Labo/* suivi de n'importe quelle chaîne de caractères s'apparie avec la cible de exemple 1.

Une condition : elle est une expression booléenne qui, au-delà de la cible de la règle, raffine davantage l'applicabilité de la règle. La plage de la ligne 17 à la ligne 25 représente la modélisation de la condition de  $E_2$ . Une règle peut ne pas contenir de condition ; dans ce cas, l'évaluation de la règle se résume à l'évaluation de sa cible.

Un effet : il est l'intention voulue par le rédacteur de la règle lorsque la règle est évaluée à *VRAI*. La règle est évaluée à *VRAI* lorsque sa cible et sa condition sont toutes évaluées à *VRAI*. L'effet prend deux valeurs qui sont : **Permit** lorsqu'il s'agit d'autoriser et **Deny** lorsqu'il s'agit d'interdire. L'effet de  $E_1$  est **Deny** et celui de  $E_2$  est **Permit**.

#### 4.1. PRÉSENTATION DE XACML

Des obligations : elles sont des actions indiquées par le rédacteur de la règle et qui doivent obligatoirement être exécutées conjointement avec l'application de l'effet de la règle. *Exemple : envoyer un courriel au patient quand un utilisateur accède à son dossier.* Le système ne peut pas appliquer la décision de l'évaluation d'une requête, s'il n'est pas en mesure de remplir les obligations qui accompagnent la décision. L'ajout des obligations dans une règle est optionnel.

Des avis : ils sont des actions indiquées par le rédacteur de la règle et qui peuvent être exécutées au moment de l'application de l'effet de la règle. A la différence des obligations, l'échec de l'exécution des avis ne bloque pas l'application de la décision.

##### La politique

Elle comprend une ou plusieurs règles, une cible, un algorithme de combinaison des règles («*rule-combining algorithm*») et optionnellement des obligations et/ou des avis.

La cible d'une politique spécifie un ensemble de requêtes auxquelles la politique s'applique. La cible de l'exemple 1 est définie dans la plage de la ligne 3 à la ligne 14. La cible de la politique peut ne pas être définie. Dans ce cas, la cible de la politique est calculée sur la base des cibles des règles que comprend la politique. La norme XACML ne fournit pas un algorithme pour faire ce calcul. Deux méthodes sont par contre suggérées pour faire le calcul. La première méthode consiste à faire une union de l'ensemble des cibles des règles contenues dans la politique et la deuxième méthode consiste à faire une intersection de l'ensemble des cibles des règles contenues dans la politique. Lorsqu'aucune cible n'est définie, alors les règles et les politiques s'appliquent à toutes les requêtes.

L'algorithme de combinaison des règles définit une procédure pour combiner les règles de la politique. Lors de l'évaluation des règles contenues dans une politique, plusieurs règles peuvent être applicables à la requête. L'algorithme de combinaison des règles permet de sélectionner une seule règle pour rendre la décision. L'algorithme **deny-overrides** est utilisé dans la politique exemple 1. Le **deny-overrides** sélectionne la première règle applicable qui a pour effet **Deny** pour rendre la décision. XACML

## CHAPITRE 4. APPLICATION DE XACML AU CAS D'ÉTUDE

définit plusieurs algorithmes de combinaison : *permit-overrides*, **first-applicable**, *only-one-applicable*, etc. En dehors des algorithmes définis dans la norme, l'utilisateur XACML peut également définir ses propres algorithmes de combinaison.

Les obligations / avis sont des actions indiquées par le rédacteur de la politique et qui sont exécutées au moment d'appliquer la décision. La norme XACML propose d'associer les obligations/ avis des règles contenues dans une politique avec les obligations/ avis de la politique.

### L'ensemble de politiques

Il comprend plusieurs politiques ou (inclusif) d'autres ensembles de politiques, une cible, un algorithme de combinaison de politiques («*policy-combining algorithm*») et optionnellement des obligations ou des avis. La cible d'un ensemble de politiques définit l'ensemble des requêtes auxquelles s'applique l'ensemble de politiques. L'algorithme de combinaison de politiques définit une procédure pour sélectionner une politique parmi celles contenues dans l'ensemble de politiques. Les obligations ou les avis sont des actions indiquées par le rédacteur de l'ensemble de politiques, et qui sont exécutées au moment d'appliquer la décision rendue par le PDP.

### Les requêtes et les réponses XACML

Les requêtes et les réponses XACML sont aussi des documents XML. Une requête contient les attributs *sujet* («*subject*»), *ressource* («*resource*») et *action*. L'attribut *sujet* définit l'auteur de la requête. L'attribut *ressource* définit la ressource que le sujet de la requête souhaite accéder. L'attribut *action* définit l'action que le sujet souhaite exécuter sur la ressource.

La réponse contient la décision rendue après évaluation de la requête. Elle peut contenir une séquence de résultats, car chaque ressource demandée a son propre résultat. La balise `<Result>` délimite chaque résultat. Elle contient les balises `<Decision>`, `<Status>` et `<Obligations>`. La balise `<Decision>` contient la décision rendue. Il y a quatre valeurs possibles qui peuvent être rendues après l'évaluation d'une requête XACML : *Permit*, *Deny*, *NotApplicable*, *Indeterminate*. Lorsque la valeur retournée est



## 4.1. PRÉSENTATION DE XACML

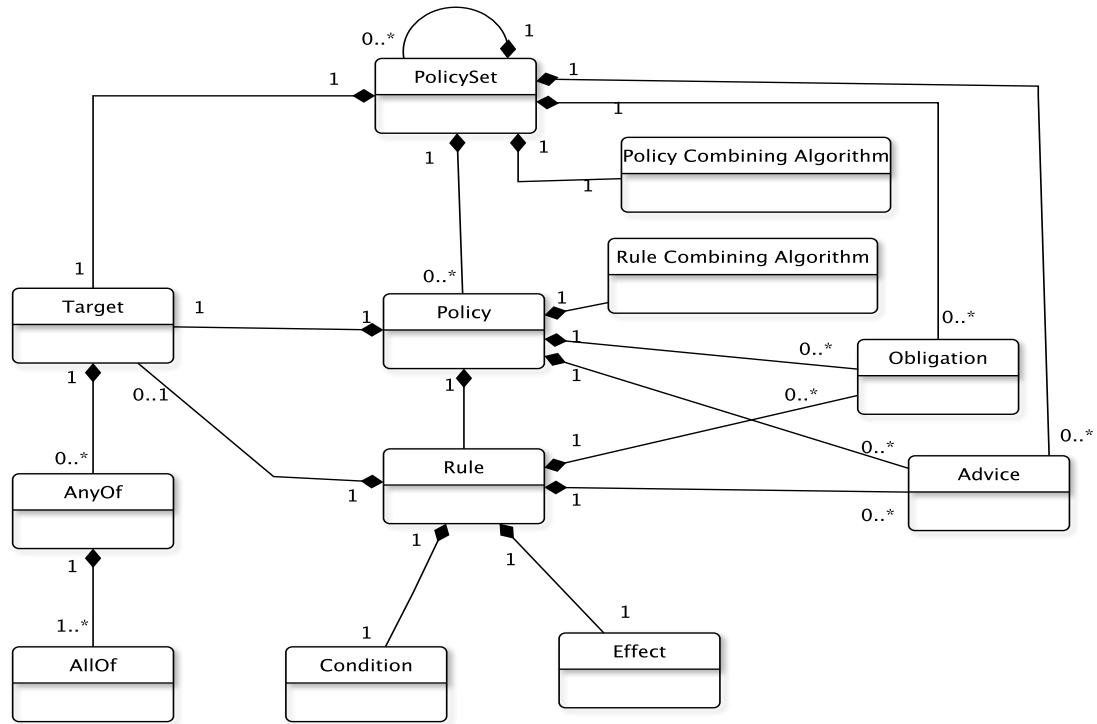


figure 4.1 – Modèle du langage XACML, tel que fourni dans la version 3 de la norme XACML

**Permit** ou **Deny**, cela signifie qu’une règle a été sélectionnée pour rendre la décision. La valeur retournée est l’effet de la règle qui a été sélectionnée. Lorsqu’aucune règle applicable à la requête n’a été trouvée, c’est la réponse **NotApplicable** qui est renvoyée. En cas d’erreur qui empêche l’évaluation de se poursuivre, c’est la valeur **Indeterminate** qui est retournée. Il y a plusieurs types d’erreurs qui peuvent survenir : connexion interrompue avec une base de données, division par zero etc. La balise *<Status>* indique l’erreur survenue lors de l’évaluation de la requête. La balise *<Obligations>* contient la liste des obligations.

### 4.1.2 Architecture du modèle de contrôle d’accès XACML

XACML ne se limite pas seulement à la spécification d’un langage pour décrire les politiques, les requêtes et les réponses. Il définit également une architecture qui

## CHAPITRE 4. APPLICATION DE XACML AU CAS D'ÉTUDE

présente les différentes entités qui doivent être implémentées pour mettre en place un système de contrôle d'accès. La figure 4.3 présente le diagramme de flux de données du modèle XACML. La description du fonctionnement de XACML se présente comme suit. Le point d'administration des politiques (PAP : «*policy administration point*») écrit les politiques et les rend disponibles au point de décision des politiques (PDP : «*policy decision point*»). Lorsqu'une requête d'accès à une ressource est envoyée, le point d'application des politiques (PEP : «*policy enforcement point*») la reçoit. Le PEP transmet la requête dans son format natif (Si la requête reçue est dans un format texte, le PEP la transmet au format texte), au gestionnaire de contexte («*Context Handler*») qui la réécrit en langage XACML. Lors de la réécriture de la requête, le gestionnaire de contexte peut avoir besoin de la valeur de certains attributs, comme par exemple la date et l'heure du système. Il demande au point d'information des politiques (PIP : «*policy information point*») qui lui retourne ces valeurs. La requête réécrite en XACML est envoyée au PDP. Le PDP détermine la politique applicable à la requête, évalue la requête et retourne la réponse au gestionnaire de contexte. La réponse envoyée par le PDP est au format XACML. Le gestionnaire de contexte convertit la réponse au format natif de la requête et l'envoie au PEP. Le PEP reçoit la réponse, applique la décision rendue conjointement avec les obligations.

### 4.1.3 Profils XACML utilisés pour modéliser le cas d'étude

XACML définit plusieurs profils qui permettent de représenter des politiques qui répondent à des exigences particulières. Nous explorons dans cette sous-section des profils que nous avons utilisés pour modéliser notre système. Ces profils sont : le profil «*Core and Hierarchical Role Based Access Control (RBAC)*» [54] et le profil «*Hierarchical Resource*» [55].

**Le profil «*core and hierarchy RBAC*»** décrit comment modéliser le «*core and hierarchy RBAC*» en XACML. Pour faire cette modélisation, XACML ne tient pas compte de la notion de session dans laquelle les rôles sont activés. Il suppose que les rôles sont activés par défaut. De même XACML ne s'occupe pas de l'assignation des rôles aux utilisateurs. Le rôle de l'utilisateur doit se trouver dans la requête qui est soumise au PDP. La norme XACML admet cependant qu'une nouvelle entité puisse

#### 4.1. PRÉSENTATION DE XACML

être implémentée pour faire l'assignation des rôles aux utilisateurs et l'activation des rôles. Elle ne donne pas une spécification pour cette nouvelle entité.

La modélisation des politiques en utilisant le profil «*core and hierarchy RBAC*», améliore RBAC car avec la balise *<condition>* présente dans les règles XACML, il devient possible de modéliser des règles dépendant d'un contexte. Nous avons montré dans les critiques de RBAC à la section 3.3 qu'il est impossible de modéliser en RBAC des règles dépendant d'un contexte.

La modélisation des politiques en utilisant le profil «*core and hierarchy RBAC*» se fait de la façon suivante : pour chaque rôle du système, comme le rôle médecin par exemple, on crée un ensemble de politiques appelé «*Role PolicySet (RPS)*». Toutes les permissions accordées au rôle médecin sont créées dans un nouvel ensemble de politiques appelé «*Permission PolicySet (PPS)*». L'assignation des permissions au rôle se fait en référant le *PPS* dans *RPS*. Le fait de référencer le *PPS* dans *RPS* entraîne l'inclusion dans *RPS*, lors de l'évaluation des requêtes, des permissions définies dans *PPS*. Lorsqu'un médecin demande à accéder à une ressource, le *RPS* définie pour le rôle médecin est sélectionnée pour évaluer la requête. Le médecin est autorisé à accéder à la ressource, s'il y a une permission incluse dans le *RPS* qui autorise l'accès à la ressource demandée.

L'héritage entre les rôles est représenté comme suit. Si un rôle *X* hérite d'un rôle *Y*, on référence le *PPS* de *Y* dans le *PPS* de *X*.

La manière utilisée en XACML pour modéliser RBAC a suscité plusieurs critiques. En effet, il est reproché à XACML de ne pas être en mesure de capturer toutes les exigences du modèle RBAC, et également de rendre la délégation de rôle impossible [25].

**Le profil hiérarchie de ressources** décrit comment XACML gère le contrôle d'accès aux ressources organisées sous forme hiérarchique. Le graphe pour représenter les ressources hiérarchiques est orienté et acyclique. Le profil hiérarchie de ressources offre un avantage dans l'écriture des règles de contrôle d'accès. En écrivant une règle qui s'applique à un nœud dans la hiérarchie de ressources, la même règle s'applique aussi aux nœuds descendants et cela évite de réécrire la règle pour ceux-ci. La spécification offerte pour implémenter le profil hiérarchie de ressources repose sur trois points :

- la représentation de l'identité d'un nœud dans une hiérarchie ;
- les requêtes d'accès aux ressources hiérarchiques ;
- la modélisation des politiques portant sur les ressources hiérarchiques.

XACML met l'accent sur le fait que le nœud dans la hiérarchie des ressources doit être représenté de la même manière dans la politique que dans la requête envoyée au PDP. La norme XACML recommande des représentations pour identifier un nœud. L'utilisateur du profil hiérarchie de ressources peut également définir sa propre représentation pourvu que le PAP et PEP utilisent la même représentation pour identifier le nœud dans la politique et dans la requête. Lorsque les ressources à protéger sont représentées par un document XML, pour les écrire dans les politiques, XACML recommande d'utiliser l'expression *XPath* pour identifier les nœuds. Dans ce cas, le document XML contenant les ressources est inclus dans les requêtes des utilisateurs. Sachant que les requêtes volumineuses sont consommatrices de la bande passante, le document doit alors être de petite taille. Lorsque les ressources ne sont pas dans un document XML, XACML recommande d'utiliser soit les URI conformément au RFC3986 [2], soit les ascendants du nœud pour l'identifier.

### 4.1.4 Processus de prise de décision

Dans l'architecture du modèle XACML, c'est le PDP qui évalue les requêtes et rend les décisions. Lorsque le PDP reçoit une requête constituée généralement des attributs : « *sujet, action et ressource* », il récupère les valeurs des attributs contenus dans la requête. Ces valeurs sont confrontées avec celles des attributs des cibles des politiques pour déterminer la politique applicable à la requête. Toutes les règles de contrôle d'accès se trouvent dans un seul document XACML qui a pour balise racine une politique ou un ensemble de politiques. Le PDP parcourt le document XACML pour rechercher la politique applicable à la requête. Ce sont les cibles des ensembles de politiques et des politiques qui sont d'abord évaluées. L'évaluation d'une cible se fait de la façon suivante : au niveau de chaque cible, il y a une fonction booléenne pour chaque attribut (ressource, action, sujet). Chaque fonction booléenne fait l'appariement entre la valeur de l'attribut dans la requête, et la valeur de l'attribut dans la cible. Exemple de fonction : `<Match MatchId="urn:oasis:names:tc:xacml:1.0:function :`

#### 4.1. PRÉSENTATION DE XACML

*string-regex-match*". Les résultats de toutes les fonctions booléennes de la cible, sont combinées par des **AllOf** et des **AnyOf** pour donner un seul résultat qui est le résultat de l'évaluation de la cible. Les **AllOf** font un **ET** logique entre les résultats des fonctions booléennes et les **AnyOf** font un **OU** logique entre les résultats des **AllOf**. Une politique est applicable lorsque le résultat de l'évaluation de sa cible est *VRAI*.

Pour illustrer le processus de prise de décision, nous prenons l'exemple des trois politiques de la figure 4.2 :  $P_1$ ,  $P_2$  et  $P_3$ . Ces trois politiques sont dans un même ensemble de politiques et l'algorithme de combinaison des politiques de cet ensemble de politiques est **deny-overrides**.

P1	P2	P3
<pre> &lt;Policy rule-combining-algorithm:deny- overrides"&gt;   &lt;Target&gt;     &lt;AnyOf&gt;       &lt;AllOf&gt;         &lt;Match function:           string-equal&gt;           &lt;resource&gt;             <b>A</b>           &lt;/resource&gt;         &lt;/Match&gt;       &lt;/AllOf&gt;     &lt;/AnyOf&gt;   &lt;/Target&gt;   &lt;Rule RuleId="rule1" Effect="Deny"&gt;     &lt;Target/&gt;   &lt;/Rule&gt; &lt;/Policy&gt; </pre>	<pre> &lt;Policy rule-combining-algorithm:deny- overrides"&gt;   &lt;Target&gt;     &lt;AnyOf&gt;       &lt;AllOf&gt;         &lt;Match function:           string-equal&gt;           &lt;resource&gt;             <b>B</b>           &lt;/resource&gt;         &lt;/Match&gt;       &lt;/AllOf&gt;     &lt;/AnyOf&gt;   &lt;/Target&gt;   &lt;Rule RuleId="rule1" Effect="Deny"&gt;     &lt;Target/&gt;     &lt;Condition&gt;       &lt;Subject&gt;         <b>Alice Fertier</b>       &lt;/Subject&gt;     &lt;/Condition&gt;   &lt;/Rule&gt;   &lt;Rule RuleId="rule2" Effect="Permit"&gt;     &lt;Target/&gt;     &lt;Condition&gt;       &lt;Subject-role&gt;         <b>infirmiere</b>       &lt;/Subject-role&gt;     &lt;/Condition&gt;   &lt;/Rule&gt; &lt;/Policy&gt; </pre>	<pre> &lt;Policy rule-combining-algorithm:deny- overrides"&gt;   &lt;Target&gt;     &lt;AnyOf&gt;       &lt;AllOf&gt;         &lt;Match function:           string-equal&gt;           &lt;Subject-role&gt;             <b>Infirmiere</b>           &lt;/Subject-role&gt;         &lt;/Match&gt;       &lt;/AllOf&gt;     &lt;/AnyOf&gt;   &lt;/Target&gt;   &lt;Rule RuleId="rule1" Effect="Permit"&gt;     &lt;Target/&gt;     &lt;Condition&gt;       &lt;resource&gt;         <b>B</b>       &lt;/resource&gt;     &lt;/Condition&gt;   &lt;/Rule&gt;   &lt;Rule RuleId="rule2" Effect="Deny"&gt;     &lt;Target/&gt;     &lt;Condition&gt;       &lt;resource&gt;         <b>C</b>       &lt;/resource&gt;     &lt;/Condition&gt;   &lt;/Rule&gt; &lt;/Policy&gt; </pre>

figure 4.2 – Codes simplifiés des politiques  $P_1$ ,  $P_2$  et  $P_3$

**Exemple de requête :** *Alice Fertier est une infirmière qui veut accéder à la ressource B.*

Au premier tour de l'évaluation faite par le PDP, il faut que l'ensemble de politiques qui contient les trois politiques  $P_1$ ,  $P_2$  et  $P_3$ , soit sélectionné et pour cela cet ensemble de politiques doit s'apparier avec la requête. Nous supposons que cette condition est satisfaite. A ce premier tour, la politique  $P_1$  est écartée, car sa cible ne s'apparie pas avec la requête. Le PDP a deux politiques  $P_2$  et  $P_3$  à évaluer.

Au deuxième tour de l'évaluation, le PDP évalue chacune des deux politiques applicables, en évaluant les règles que chacune d'elle contient. Chaque politique évaluée a une valeur qui est celle issue de la combinaison de l'ensemble des résultats de l'évaluation des règles qu'elle contient. Une règle est applicable à la requête si sa cible s'apparie avec la requête et aussi que la requête respecte la condition définie dans la règle. Dans  $P_2$  et  $P_3$ , il n'y a pas de cible définie. Par conséquent, les cibles des règles sont égales à celles des politiques qui les contiennent. Dans l'évaluation de  $P_3$ , il y a une seule règle qui est applicable à la requête : la règle *rule1* qui a pour effet = **Permit**. Le PDP retourne **Permit** comme valeur de l'évaluation de la politique  $P_3$  ( $P_3 = \text{Permit}$ ).

Dans l'évaluation de la politique  $P_2$ , les deux règles sont applicables à la requête. Les cibles que les deux règles contiennent ainsi que les conditions s'apparient avec la requête. Pour sélectionner une seule règle parmi les deux, le PDP fait usage de l'algorithme de combinaison des règles de  $P_2$ . Dans  $P_3$ , comme il n'y avait qu'une seule règle applicable, le PDP n'a pas fait usage de l'algorithme de combinaison des règles de  $P_3$ . L'algorithme de combinaison des règles de  $P_2$  est **deny-overrides**, ce qui fait que c'est la règle *rule1* qui a l'effet **Deny**, qui est choisie. La valeur de  $P_2$  est donc **Deny** ( $P_2 = \text{Deny}$ ).

Au troisième tour de l'évaluation, le PDP utilise l'algorithme de combinaison des politiques de l'ensemble de politiques, pour choisir une seule politique parmi  $P_2$  et  $P_3$ . La valeur de  $P_2$  est **Deny** et celle de  $P_3$  est **Permit**. L'algorithme de combinaison des politiques de l'ensemble de politiques est **deny-overrides**. Par conséquent, c'est la valeur **Deny** de  $P_2$  qui est choisie comme valeur de l'ensemble de politiques. Il n'a pas d'autres ensembles de politiques, dans lesquels est contenu l'ensemble de politiques de notre exemple, pour que le PDP combine sa valeur avec les résultats d'autres évaluations. Le PDP retourne alors au PEP la valeur **Deny** qui est la décision finale de l'évaluation

## 4.2. MISE EN OEUVRE DE XACML

de la requête de Alice Fertier.

## 4.2 Mise en oeuvre de XACML

La mise en oeuvre de XACML consiste à implémenter un système de contrôle d'accès qui respecte les exigences décrites dans la norme XACML. Il y a plusieurs implementations de la norme XACML : sunXACML[52], Balana[58], etc. Notre choix s'est porté sur l'implémentation Balana qui couvre toutes les versions actuelles de la norme. Balana est une implémentation open source qui intègre aussi un éditeur de politiques XACML. Dans les sous-sections qui suivent, nous présentons d'abord Balana, ensuite nous décrivons le fonctionnement du système que nous avons implémenté et enfin nous tirons une conclusion sur le processus de mise en oeuvre de XACML.

### 4.2.1 Balana

L'implémentation Balana de la norme XACML est utilisée pour tester les politiques du cas d'étude. L'implémentation Balana est basée sur celle de SUN (sunXACML). Balana 1.0 est utilisée dans cette étude. Elle comporte plusieurs modules dont :

- **Core Balana** : implémente toutes les fonctions exécutées par le PDP pour rendre une décision.
- **Balana Samples** : contient des exemples d'utilisation.
- **Balana Utils** : selon la description offerte, ce module contient des utilitaires tels qu'un éditeur de politiques XACML. Balana Utils n'est pas intégré dans la version de Balana que nous avons utilisée. L'éditeur de politiques utilisé est *UMU-XACML-Editor* [3], développé à Université de Murcia (Espagne).
- **Balana distribution** : conteneur de la documentation, des jars.

Balana implémente le moteur XACML sur la base des spécifications de la norme. Le soin est laissé à celui qui veut utiliser Balana d'implémenter le PEP, le gestionnaire de contexte et le PIP. Ce sont ces trois entités que nous avons implémentées pour ajouter au noyau de Balana afin de constituer notre système. Un module appelé **CHUS** a été créé avec comme référence le module *core Balana* pour permettre l'importation des classes implémentant les fonctions du PDP. Le module **CHUS**

contient les implémentations du PEP, du gestionnaire de contexte et du PIP, qui sont faites en fonction des besoins de notre cas d'étude.

### 4.2.2 Fonctionnement du système implémenté

Les communications entre les différentes entités implémentées dans notre système se font à l'image de celles décrite par le flux de données XACML de la figure 4.3. Nous reconsidérons l'exemple 1 de la section 4.1.1 pour illustrer ce fonctionnement.

**Requête :** *l'infirmière Alice Fertier souhaite accéder au VIH\_123 de Romain.*

La description du fonctionnement de notre système est faite par entité de l'architecture du modèle XACML.

**Le PAP :** les politiques sont créées et mises à jour par le PAP. Le PAP rend également les politiques créées disponible au PDP (lien 1 de la figure 4.3). Nous avons utilisé l'éditeur *UMU-XACML-Editor* pour écrire les politiques en XACML. Cet éditeur joue donc le rôle du PAP. Les politiques écrites sont enregistrées dans un dossier appelé *policies*. Le chemin d'accès à ce dossier est indiqué au PDP. Les paragraphes qui suivent décrivent comment la politique de l'exemple 1 a été écrite en XACML.

Pour écrire en XACML la politique de l'exemple 1, nous avons choisi d'indiquer dans la cible de la politique, la ressource sur laquelle porte cette politique, à savoir les résultats de laboratoires de Romain. Cette politique ne peut être choisie par le PDP que lorsque la requête porte sur un laboratoire de Romain. La valeur de la ressource est : *Rom\_001/Labo/\**, en notation expressions régulières. *Rom\_001* représente l'identifiant de Romain et *Labo* représente l'identifiant du type de donnée laboratoire.

La politique comprend deux règles correspondant aux deux exigences du scénario. Les deux règles n'ont pas de cible définie. La cible de chacune d'elle correspond à celle de la politique. Chacune de ces deux règles comporte une condition. Pour la première règle *rule1*, la condition porte sur le sujet *Alice Fertier*. Pour la deuxième règle *rule2*, la condition porte sur le rôle *infirmière*. La fonction booléenne **any-of** a été utilisée pour représenter les deux conditions.



## 4.2. MISE EN OEUVRE DE XACML

La fonction **any-of** prend  $n + 1$  arguments avec  $n \geq 1$ . Le premier argument est une fonction booléenne, ici **string-equal** qui compare deux chaînes de caractères. Les  $n$  arguments restant se répartissent comme suit :  $n - 1$  valeurs d'un type primitif et un tableau de valeurs de type primitif. L'évaluation faite par la fonction **any-of** se passe de façon suivante : chacune des  $n - 1$  valeurs est comparée par la fonction booléenne **string-equal** passée en paramètre avec chacune des valeurs du tableau passé également en paramètre. Les résultats intermédiaires sont combinés par un *OU* logique et le résultat de la combinaison retournée par la fonction **any-of**. Prenons l'exemple de l'utilisation de la fonction *any-of* dans la condition de *rule2*. Nous avons en arguments de la fonction **any-of** : la fonction booléenne **string-equal**, la valeur *infirmière* et un tableau contenant l'ensemble des rôles du sujet *Alice Fertier*. Le tableau n'est pas une constante déclarée, il est généré par le PIP lors de l'évaluation de la requête. Pour générer ce tableau, le PDP sait à travers la balise *<AttributeDesignator>* que c'est le rôle du sujet qui est attendu comme argument de la fonction **any-of**. Alors, le PDP envoie une requête au PIP qui lui retourne dans un tableau l'ensemble des rôles du sujet. Si parmi les éléments retournés dans le tableau il y a la chaîne *infirmière*, alors la condition de *rule1* est satisfaite.

**Le PEP et le gestionnaire de contexte :** ce sont deux entités différentes dans l'architecture XACML, mais il arrive par abus de langage qu'on désigne ces deux entités par la seule expression PEP. Ces deux entités sont implémentées dans notre système par la classe *Health.java*. Le code de cette classe est fourni en annexe [A](#).

Le PEP reçoit les requêtes des utilisateurs et le gestionnaire de contexte réécrit la requête en XACML. La requête peut être envoyée sous plusieurs formats. Dans les exemples fournis avec l'implémentation Balana, une console se crée au lancement du programme et permet à l'utilisateur de saisir son identifiant et la ressource qu'il veut accéder. Les données saisies sont passées à la méthode *createXACMLRequest()* qui joue le rôle de gestionnaire de contexte.

Dans notre implémentation du PEP et du gestionnaire de contexte, nous avons permis que les requêtes soient directement récupérées en XACML. Notre choix de procéder ainsi répond à un besoin de faciliter nos tests.

La requête qui arrive étant déjà en XACML, le gestionnaire de contexte va juste

## CHAPITRE 4. APPLICATION DE XACML AU CAS D'ÉTUDE

l'ouvrir et réécrire la valeur de la ressource en respectant la représentation établie pour désigner les ressources. Nous reviendrons sur cette représentation dans la section 4.3.1. Le gestionnaire de contexte fait appel à la fonction *findResource()* qui se connecte à la base de données et retourne les informations nécessaires pour réécrire la ressource en respectant la représentation établie. La requête qui est envoyée au PEP est présentée au programme 4.2. Avant de passer cette requête au PDP, elle est réécrite et on obtient le résultat présenté au programme 4.3.

programme 4.2 – Format XACML de la requête d'Alice Fertier

```
<Request>
  <Attributes Category="access-subject" >
    <Attribute IncludeInResult="false" AttributeId="subject-id">
      <AttributeValue >Alice Fertier </AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="resource">
    <Attribute IncludeInResult="true" AttributeId="resource-id">
      <AttributeValue >VIH_123</AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="action">
    <Attribute IncludeInResult="false" AttributeId="action-id">
      <AttributeValue >accéder</AttributeValue>
    </Attribute>
  </Attributes>
</Request>
```

**Le PDP :** Le PDP est implémenté dans le module «*core BalanaI*». Aucune modification n'a été apportée au code écrit dans ce module. Pour évaluer la requête, le PDP choisit dans la liste des politiques mises à sa disposition, celle qui est applicable en se basant d'abord sur l'évaluation des cibles. Le processus de prise de décision est déjà décrit dans la sous-section 4.1.4. La politique prise dans l'exemple 1 de la section 4.1.1 est applicable lorsque la fonction **string-regexp-match** utilisée dans la cible retourne *VRAI*. Cette fonction compare la valeur de la ressource indiquée dans la politique avec la valeur de la ressource contenue dans la requête. La ressource dans la cible de la politique a pour valeur *Rom\_001/Labo/\**. Cette valeur s'apparie avec toutes les ressources des requêtes commençant par *Rom\_001/Labo/*, suivies de n'importe quelle

## 4.2. MISE EN OEUVRE DE XACML

programme 4.3 – Format XACML de la requête d’Alice Fertier réécrite

```
<Request >
  <Attributes Category="access-subject" >
    <Attribute IncludeInResult="false" AttributeId="subject-id">
      <AttributeValue >Alice Fertier </AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="resource">
    <Attribute IncludeInResult="true" AttributeId="resource-id">
      <AttributeValue >Rom_001/Labo/sang/VIH/VIH_123</AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="action">
    <Attribute IncludeInResult="false" AttributeId="action-id">
      <AttributeValue >accéder</AttributeValue>
    </Attribute>
  </Attributes>
</Request>
```

autre chaîne de caractères, d’où l’appariement avec la ressource dans la requête qui est : *Rom\_001/Labo/sang/VIH/VIH\_123*.

La politique choisie pour évaluer la requête contient deux règles se rapportant aux deux exigences du scénario. Les cibles de ces deux règles ne sont pas définies. Dans ce cas XACML dit que la cible de la politique est la même que celle des règles. Les deux règles sont alors applicables à la requête en considérant leurs cibles. XACML permet de définir dans les règles, des conditions pour raffiner davantage le choix de la règle applicable. Les deux règles comportent toutes des conditions. La condition de la première règle indique que le sujet doit être Alice Fertier. La condition de la deuxième règle indique que sujet doit être infirmière. L’évaluation de la première règle est triviale, car la requête vient avec l’*id-subject*. Pour la deuxième règle, le PDP envoie un message au PIP par sa méthode *findAttribute()*. Le PIP se connecte à la base de données où sont enregistrées les informations sur les sujets et retourne tous les rôles auxquels appartient le sujet. Dans notre exemple, Alice Fertier remplit toutes les conditions des deux règles, car elle est aussi infirmière. Les conditions n’ont pas permis d’avoir une seule règle. Le PDP utilise alors l’algorithme de combinaison de règles de la politique pour déterminer une seule règle à utiliser. L’algorithme de combinaison de règles de la politique est **deny-overrides**. Il s’en suit que c’est la règle dont l’effet

est **Deny** qui est choisie par le PDP. Le résultat retourné par le PDP en réponse à la requête de Alice Fertier est **Deny**. Toute autre infirmière différente de Alice fertier qui envoie la même demande est autorisée. Cette réponse est présentée au programme 4.4.

programme 4.4 – Exemple de réponse XACM

```
<Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
<Result>
  <Decision>Deny</Decision>
  <Status></Status>
  <Obligations></Obligations>
</Result>
</Response>
```

**Le PIP :** Balana ne donne pas une implémentation standard le PIP, le PEP et le gestionnaire de contexte. Il y a certes des exemples à l'intérieur de Balana qui montrent son fonctionnement, et dans lesquels des exemples de PEP, de PIP et de gestionnaire de contexte ont été implantés, mais l'utilisateur de Balana n'est pas tenu par ces exemples. Nous avons créé la classe *AttributFinderHealth.java* pour implémenter les fonctions du PIP. Le code de cette classe est fourni à l'annexe B.

Le PDP envoie un message au PIP via la méthode *findAttribute()* pour obtenir des compléments d'informations, comme trouver le rôle de Alice Fertier dans notre exemple. Le PIP, grâce à la méthode *findRole()* de la classe qui l'implémente, retourne au PDP tous les rôles dont le sujet Alice Fertier est membre. Cette méthode établit la connexion avec la base de données d'où sont enregistrées les informations sur les sujets.

### 4.2.3 Conclusion de la mise en oeuvre de XACML

Les tâches à accomplir pour utiliser Balana dans un système de contrôle d'accès XACML se résument comme suit :

- implémenter le PEP qui reçoit les requêtes,
- implémenter le gestionnaire de contexte pour la réécriture des requêtes en XACML,
- implémenter le PIP qui indique comment trouver les valeurs des attributs,

### 4.3. MODÉLISATION DU CAS D'ÉTUDE EN XACML

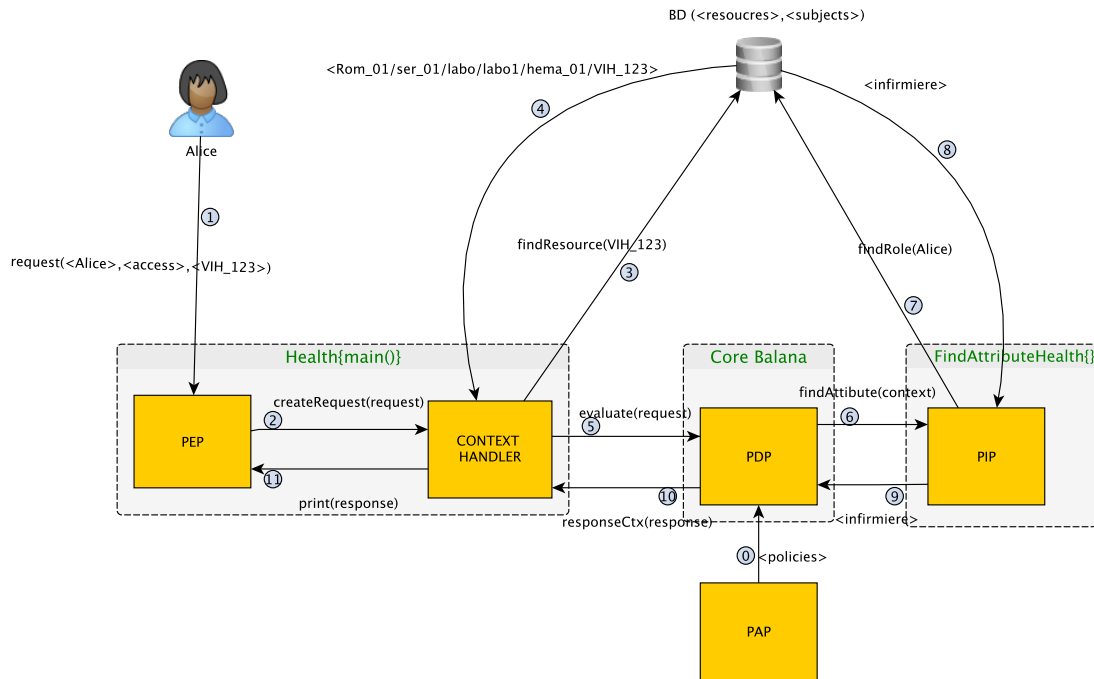


figure 4.3 – Diagramme de flux de données du système XACML

- implémenter soi-même son PAP pour écrire les politiques ou utiliser un éditeur existant.

### 4.3 Modélisation du cas d'étude en XACML

La modélisation du cas d'étude consiste à écrire en langage XACML, les politiques du cas d'étude. Le graphe des ressources de notre système est orienté et acyclique, ce qui est conforme aux structures hiérarchiques décrites dans le profil hiérarchie de ressources de XACML. Les professionnels travaillant à l'hôpital constituent les sujets du système. Ils sont organisés en plusieurs catégories : fonctions, services, spécialités, etc. La figure 2.2 présente le graphe des sujets.

Les informations sur les ressources et les sujets sont enregistrées dans une même base de données PostgreSQL. La figure 4.4 présente le modèle logique de la base de données contenant les informations sur les sujets et les ressources.

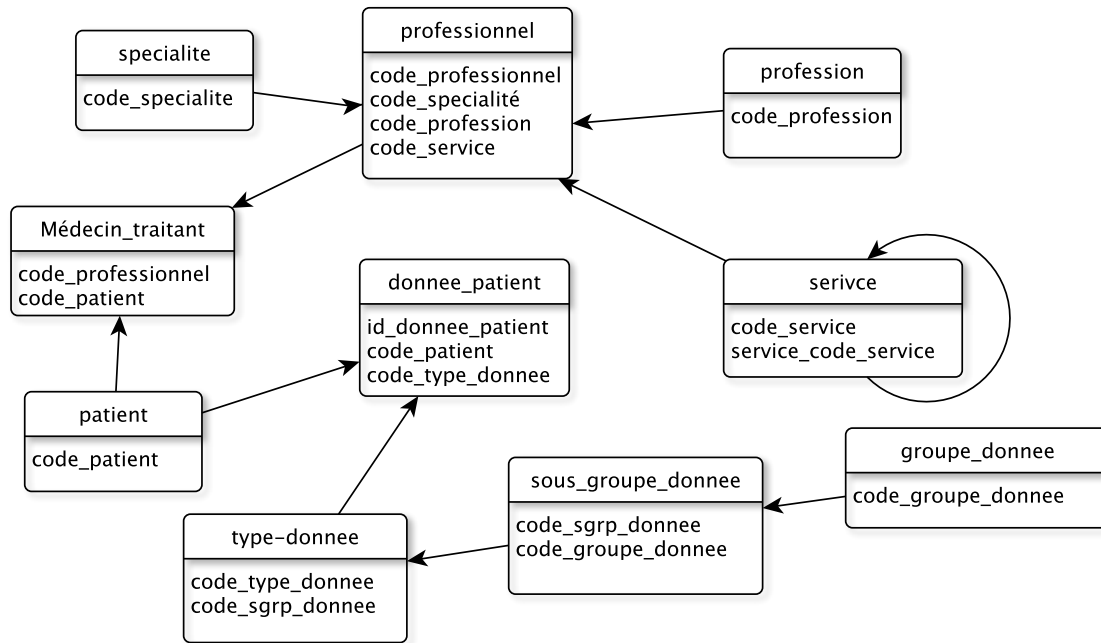


figure 4.4 – Modèle logique de données de la base de données du système XACML

Les règles qui forment les politiques de contrôle d'accès dans notre cas d'étude, émanent de plusieurs sources :

- la loi,
- l'hôpital,
- les patients.

La loi et l'hôpital définissent des règles qui s'appliquent aux dossiers d'un groupe de patients ou de l'ensemble des patients. Le patient définit des règles qui s'appliquent uniquement à ses propres dossiers. Les règles sont contenues dans les politiques selon le modèle du langage XACML. Le document XACML contenant toutes les règles a une seule racine : la balise `<policySet>` ou la balise `<policy>`. Nous devons alors créer une seule politique et y mettre toutes les règles, ou bien nous créons un ensemble de politiques pour contenir d'autres ensembles de politiques et des politiques. En adoptant le profil «*core and hierarchical RBAC*», le choix de créer un ensemble de politiques pour contenir toutes les politiques s'impose, car comme décrit à la section 4.1.3, l'utilisation du profil «*core and hierarchical RBAC*», requière la création de

### 4.3. MODÉLISATION DU CAS D'ÉTUDE EN XACML

plusieurs ensembles de politiques. Plusieurs autres raisons justifient le choix de créer un ensemble de politiques pour contenir toutes les politiques :

- Faciliter l'administration des règles : le choix de l'ensemble de politiques comme conteneur principal des politiques, nous permet de créer à l'intérieur de ce conteneur d'autres ensembles de politiques et des politiques. On peut regrouper les règles suivant leurs cibles et cela rend le document de politiques lisible et facilite l'administration des règles.
- Délimiter le champ d'application des algorithmes de combinaison : les algorithmes de combinaison permettent de sélectionner une seule règle ou une seule politique pour évaluer une requête. Si nous avons choisi de mettre toutes les règles dans une seule politique, nous aurions utilisé un seul algorithme de combinaison de règles pour tout notre système, et nous ne pourrions pas modéliser à la fois, le fait qu'un patient dise qu'en cas de conflit entre ses règles la préséance est donnée à la règle qui interdit, et un autre patient qui voudrait que la préséance soit accordée à la règle qui autorise. Le regroupement que l'ensemble de politiques nous permet de faire, nous permet également de choisir l'algorithme de combinaison approprié pour chaque type de regroupement.

Il reste maintenant à savoir s'il faut écrire une politique par type de données, une politique par article de la loi, une politique par sujet, ou une politique par patient.

Définir une politique par type de données revient à indiquer le type de données dans la cible de la politique. Toutes les règles des patients et de la loi sur le type de données doivent figurer dans la politique définie. Par exemple, on peut définir une politique pour toutes les données des laboratoires de tous les patients, en précisant cette valeur dans la catégorie ressource de la cible :

```
<AttributeValue DataType=« http://www.w3.org/2001/XMLSchema#string »>  
*/Labo/*</AttributeValue>.
```

L'option d'écrire les politiques par type de données n'est pas appropriée. Elle entraîne une multiplication des règles qui peut rendre l'administration difficile. Lorsqu'un patient dit par exemple qu'il interdit l'accès à l'ensemble de ses données (de tout type), il faudra créer autant de règles qu'il y a de type de données pour représenter cette exigence du patient.

Écrire les politiques par sujet, c'est-à-dire pour chaque professionnel, créer une

politique qui contient toutes ses autorisations et ses interdictions, est une option qui n'est pas à envisager en XACML. Si un patient décide d'une règle qui touche tous les sujets, il faut répéter cette règle autant de fois qu'il y a de sujets. Cette option conduit à une multiplication des règles et rend par conséquent l'administration difficile.

L'option adoptée est celle qui consiste à créer une politique par patient. Chaque politique d'un patient donné contient toutes les règles définies par ce patient. Toutes les règles qui concernent les données de plusieurs patients comme les lois sont représentées par des politiques à part, et elles ne sont pas dupliquées à l'intérieur de chaque politique de patient.

### 4.3.1 Représentation des ressources dans notre système

Les ressources dans notre système sont organisées sous forme hiérarchique comme le montre la figure 2.3. Les règles d'accès comportent naturellement de l'héritage (une règle sur tous les tests VIH, une règle sur tous les examens psychiatriques, etc). Une requête XACML porte sur une ressource spécifique, exemple le test *VIH\_123*. Lors de l'évaluation d'une requête, le PDP doit apparier une règle portant sur une ressource de la hiérarchie, avec une requête portant sur une instance particulière d'un type de ressource de la hiérarchie.

Nous avons vu à la section 4.1.3 que XACML offre le profil hiérarchie de ressources qui permet de modéliser les règles portant sur des ressources organisées sous forme hiérarchique. En optant pour ce profil, nous avons le choix de définir notre propre représentation pour identifier un nœud dans la hiérarchie, ou bien utiliser l'une des représentations recommandées. En optant pour une des représentations recommandées, nous avons le choix entre les expressions XPath, les URI et les identifications sur la base des nœuds ascendants.

Les expressions *XPath* s'appliquent aux documents XML, ce qui n'est pas le cas dans notre système. Les identifications des nœuds par les URI ou sur la base des nœuds ascendants sont utilisables dans notre système.

L'utilisation des URI et les nœuds ascendants requiert un ordre entre les colonnes qui entrent dans la désignation des ressources dans la base de données. On pourra ainsi dire que telle colonne est l'ancêtre de telle autre (tel nœud est l'ancêtre de tel



### 4.3. MODÉLISATION DU CAS D'ÉTUDE EN XACML

autre). Les ressources dans notre système sont représentées suivant cet ordre : **patient/groupe\_donnees/sgrp\_donnees/type\_donnee/id\_donnee**.

XACML précise dans le profil hiérarchie des ressources que la ressource dans la requête envoyée au PDP et la ressource dans la règle doivent avoir la même représentation. Si par exemple dans l'ordre que nous avons établi il y a une inversion des colonnes entre la représentation de la ressource dans la règle et la représentation de la ressource dans la requête, la correspondance n'aura jamais lieu.

Lorsqu'un sujet envoie une requête, il utilise seulement la dernière colonne dans l'ordre qui est : *id\_donnee*. Il ne connaît pas les autres colonnes. C'est le gestionnaire de contexte qui, dès qu'il reçoit une requête, se connecte à la base de données, récupère les valeurs des autres colonnes, réécrit la ressource en respectant l'ordre avant d'envoyer la requête au PDP.

Dans la modélisation du cas d'étude, nous avons plus utilisé les URI. Les expressions régulières ont également été utilisées pour écrire les URI. Ainsi :

*Rom\_001/\** : désigne l'ensemble des ressources appartenant au patient dont l'identifiant est *Rom\_001*. *Rom\_001/Labo/\** : désigne tous les résultats de laboratoires du patient dont l'identifiant est *Rom\_001*.

L'utilisation des expressions régulières permet de modéliser la hiérarchie des ressources. Elle évite d'énumérer toutes les ressources appartenant à une catégorie donnée pour modéliser une politique, elle permet de prendre en compte les nouvelles données dans les politiques préalablement définies. Les sous-sections qui suivent présentent la modélisation de chaque scénario du cas d'étude.

#### 4.3.2 Scénario 1

*Patrice est un patient qui souhaite interdire l'accès à l'ensemble de ses données personnelles (dépistage du VIH, test ADN, Examens psychiatriques) à l'ensemble des professionnels de l'hôpital.*

Notion : Traitement d'une seule règle

#### Solution du scénario 1

**Les exigences du scénario 1**  $E_1$  : *Patrice est un patient qui souhaite interdire l'accès à l'ensemble de ses données personnelles à l'ensemble des professionnels de l'hôpital.* La solution consiste à créer à l'intérieur de l'ensemble de politiques, une politique dont la cible est présentée dans le programme 4.5.

programme 4.5 – Modélisation de la cible du scénario 1

```
<Target>
  <AnyOf>
    <AllOf>
      <Match MatchId="function:string-regex-match">
        <AttributeValue> Pat\_004/*</AttributeValue>
        <AttributeDesignator Category="resource"
          AttributeId="resource-id"/>
      </Match>
    </AllOf>
  </AnyOf>
</Target>
```

Dans le programme 4.5, *Pat\_004* représente l'identifiant de Patrice dans le système. Cette politique sera sélectionnée uniquement pour toute demande d'accès à n'importe quelle ressource de Patrice. La politique créée contient une seule règle :

*<Rule Effect="Deny" RuleId="exigence1"/>*

Les tests effectués après la modélisation montrent que toutes les requêtes d'accès aux dossiers de Patrice sont interdites (réponse= **Deny**).

### 4.3.3 Scénario 2

*Alain est un patient qui souhaite autoriser l'accès à l'ensemble de ses données personnelles (dépiage du VIH, test ADN, Examens psychiatriques) à l'ensemble des professionnels de l'hôpital. Cela dit, il y a une loi qui précède la règle d'Alain et qui interdit l'accès aux examens psychiatriques de tous les patients.*

Notion : Gestion de la priorité entre deux règles.

### Solution du scénario 2

#### Les exigences du scénario2

### 4.3. MODÉLISATION DU CAS D'ÉTUDE EN XACML

1.  $E_1$  : la loi interdit l'accès aux examens psychiatriques de tous les patients.
2.  $E_2$  : Alain souhaite autoriser l'accès à l'ensemble de ses données personnelles (dépistage du VIH, test ADN, Examens psychiatriques) à l'ensemble des professionnels de l'hôpital.
3.  $E_3$  : la loi est prioritaire par rapport aux règles du patient

La solution au scénario 2 consiste à :

- Créer une politique pour modéliser l'exigence  $E_1$  : il est dit précédemment que lorsqu'une règle émanant de la loi concerne tous les patients, une politique sera créée pour modéliser cette règle. Cela évite de répéter la règle dans toutes les politiques des patients. La politique créée et qui a pour identifiant *PolicyId* = « *Loi\_Sur\_Examens\_Psychiatriques* », concerne tous les examens psychiatriques des patients. La cible de la politique est définie sur le groupe de données psychiatriques dont l'identifiant est : *SER\_002*. La politique de la loi est présentée dans le programme 4.6.

programme 4.6 – Politique contenant les règles de la loi

```
<Policy PolicyId="Loi_Sur_Examens_psychiatriques"
  RuleCombiningAlgId="deny-overrides" Version="1.0">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-regex-match">
          <AttributeValue >\\*/SER_002/*</AttributeValue>
          <AttributeDesignator Category="resource"
            AttributeId="resource-id"/>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
  <Rule Effect="Deny" RuleId="exigence1"/>
</Policy>
```

La notation *\*/SER\_002/\** permet de sélectionner toutes les ressources contenant la chaîne de caractères */SER\_002/*.

La politique *Loi\_Sur\_Examens\_Psychiatriques* contient une seule règle dont l'effet est **Deny**. Lorsque cette politique est sélectionnée pour être appliquée à une requête, c'est l'effet de la règle qui sera retourné comme résultat.

## CHAPITRE 4. APPLICATION DE XACML AU CAS D'ÉTUDE

- Modéliser l'exigence  $E_2$  : une règle est créée à l'intérieur de la politique d'Alain pour modéliser cette exigence. Le programme 4.7 présente la politique d'Alain.

programme 4.7 – Modélisation exigence 2 du scénario 2

```
<Policy PolicyId="scenario2"
  RuleCombiningAlgId="permit-overrides" Version="1.0">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-regex-match">
          <AttributeValue>Aln\_001/*</AttributeValue>
          <AttributeDesignator Category="resource"
            AttributeId="resource-id"/>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
  <Rule Effect="Permit" RuleId=" exigence1"/>
</Policy>
```

- Modéliser la priorité de la loi ( $E_3$ ) : pour modéliser la priorité de la loi, nous utilisons l'algorithme de combinaison des politiques de l'ensemble de politiques. Lorsqu'un sujet demande à accéder à un examen psychiatrique d'Alain, par exemple *PSY\_001*, après transformation pour appliquer la représentation adoptée à la sous-section 4.3.1, la requête qui est envoyée au PDP a pour valeur de ressource :

*Aln\_001/SER\_002/SG\_001/DT\_003/PSY\_001*. Cette valeur de la ressource demandée s'apparie avec la ressource dans la politique d'Alain et dans la politique de la loi. En choisissant pour l'ensemble de politiques, l'algorithme de combinaison de politiques **first-applicable**, c'est la première politique applicable rencontrée au cours du traitement qui est choisie pour l'évaluation. Les deux politiques étant dans le même ensemble de politiques, considérant que la lecture du contenu du fichier des politiques se fait séquentiellement, c'est la politique applicable qui est placée en premier, qui est sélectionnée. La politique de la loi est mise au-dessus de celle d'Alain. Toutes les politiques des patients doivent être écrites en dessous de celle de la loi à l'intérieur de l'ensemble de politiques. Si par erreur la politique d'un patient est écrite au-dessus de celle de la loi,

### 4.3. MODÉLISATION DU CAS D'ÉTUDE EN XACML

les examens psychiatriques de ce patient seront accessibles et la loi sera violée. L'algorithme de combinaison de politiques choisi n'est pas le seul qui peut être utilisé ici. On pouvait par exemple utiliser **deny-overrides**. Avec l'algorithme **deny-overrides**, c'est la première règle d'interdiction applicable à la requête, qui est sélectionnée pour rendre la décision. Il y a quand même une précaution à prendre en compte dans le choix de l'algorithme de combinaison de politiques. En effet, toutes les politiques contenues dans un ensemble de politiques donné, sont soumises à l'algorithme de combinaison de politiques de cet ensemble de politiques, d'où la nécessité d'examiner les effets de l'algorithme de combinaison sur les politiques avant de l'utiliser. Les algorithmes de combinaison de XACML ne permettent pas de résoudre le conflit entre deux règles en se basant sur la priorité. Nous verrons dans nos critiques de XACML que si un attribut de priorité prend une valeur chiffrée dans chaque politique, alors nous n'aurions pas à nous inquiéter pour modéliser la priorité de la loi.

**Tests après modélisation :** Toutes les requêtes d'accès aux examens psychiatriques d'Alain, tout comme aux examens psychiatriques des autres patients, sont interdites. Toutes les requêtes d'accès aux autres données d'Alain sont autorisées.

#### 4.3.4 Scénario 3

*Romain est un patient qui souhaite interdire l'accès aux Laboratoires à l'infirmière Alice Fertier. Cela dit, Romain avait déjà une directive qui autorisait toutes les infirmières à accéder à ses Laboratoires.*

Notion : Hiérarchie des sujets.

**Solution du scénario3** Ce scénario est l'exemple pris pour expliquer le fonctionnement du système.

#### 4.3.5 Scénario 4

*Romain est un patient qui possède dans son dossier deux examens de laboratoire : laboratoire1 et laboratoire 2. Il spécifie une règle, autorisant les médecins à accéder à*

## CHAPITRE 4. APPLICATION DE XACML AU CAS D'ÉTUDE

ses laboratoires. Romain ajoute ensuite à son dossier un autre laboratoire : *laboratoire3*. Le Dr. Bernard Lapointe lance une requête pour accéder au *laboratoire3* de Romain.

Notion : Nouvelles données.

### Solution du scénario 4 :

**Les exigences du scénario 4 :**  $E_1$  : Romain autorise les médecins à toujours accéder à ses résultats de laboratoires y compris les nouveaux laboratoires.

La cible de la politique qui contrôle les accès aux données de Romain, est présentée dans le programme 4.8

programme 4.8 – cible du scenario 4

```
<Target>
  <AnyOf>
    <AllOf>
      <Match MatchId="string-regex-match">
        <AttributeValue >Rom\_002/*</AttributeValue>
        <AttributeDesignator Category="resource"
          AttributeId="resource-id"/>
      </Match>
    </AllOf>
  </AnyOf>
</Target>
```

Pour modéliser l'exigence  $E_1$ , une règle est créée à l'intérieur de la politique de Romain. La règle contient une cible qui précise qu'elle s'applique aux résultats de laboratoires de Romain. La règle contient également une condition qui limite l'accès uniquement aux professionnels médecins. Le programme 4.9 présente l'exigence  $E_1$  écrite en XACML.

L'utilisation des expressions régulières pour décrire la ressource (*Rom\_002/Labo/\**) évite de citer tous les résultats de laboratoires de Romain. Les futurs résultats de laboratoires sont pris en compte dans la règle.

**Tests après modélisation :** lorsque Pierre Bertrand, qui est médecin, envoie une demande d'accès au laboratoire *RA\_001* de Romain, la réponse est **Permit**. Lorsque Alice Fertier qui est infirmière envoie une requête d'accès à la même donnée *RA\_001*,

### 4.3. MODÉLISATION DU CAS D'ÉTUDE EN XACML

programme 4.9 – Modélisation de l'exigence 1 du scénario 4

```
<Rule RuleId="exigence1" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-regex-match">
          <AttributeValue>Rom\_002/Labo/*</AttributeValue>
          <AttributeDesignator
            Category="resource"
            AttributeId="resource-id"/>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
  <Condition>
    <Apply FunctionId="any-of">
      <Function FunctionId="string-equal"/>
      <AttributeValue>medecin</AttributeValue>
      <AttributeDesignator
        Category="access-subject"
        AttributeId="subject:role"/>
    </Apply>
  </Condition>
</Rule>
```

la réponse est **NotApplicable**. Toute requête d'accès à n'importe quelle donnée de Romain autre que ses résultats de laboratoires reçoit la réponse **NotApplicable** même si le sujet est un médecin.

#### 4.3.6 Scénario 5

*Simon est un patient qui a spécifié deux règles, la première interdit aux professionnels travaillant à l'urgence d'accéder à son dossier et la seconde autorise les professionnels travaillant en hématologie d'accéder au dossier. Le Dr. Pierre Bertrand souhaite accéder au dossier de Simon.*

Notion : Multi-appartenance

**Solution du scénario 5 :**

**Les exigences du scénario 5 :**

1.  $E_1$  : *Simon interdit d'abord les professionnels travaillant à l'urgence d'accéder à son dossier*
2.  $E_2$  : *Simon autorise ensuite les professionnels travaillant en hématologie d'accéder au dossier*

La solution au scénario 5 consiste à :

- Créer une politique en précisant dans sa cible, qu'elle s'applique aux données de Simon. La politique contient deux règles :  $E_1$  et  $E_2$ .
- Créer une règle pour modéliser l'exigence  $E_1$ . Cette règle a pour effet **Deny** et elle possède une condition qui précise qu'elle s'applique aux professionnels travaillant à l'urgence. Le programme 4.10 présente l'exigence  $E_1$  écrite en XACML.

programme 4.10 – Modélisation de l'exigence 1 du scénario 5

```
<Rule RuleId="exigence1" Effect="Deny">
  <Target/>
  <Condition>
    <Apply FunctionId="function:any-of">
      <Function FunctionId="string-equal"/>
      <AttributeValue>urgence</AttributeValue>
      <AttributeDesignator
        Category="access-subject"
        AttributeId="role"/>
    </Apply>
  </Condition>
</Rule>
```

- Créer une règle pour modéliser l'exigence  $E_2$ . La règle a pour effet **Permit**. Elle possède une condition qui précise qu'elle s'applique aux professionnels travaillant en hématologie. Le programme 4.11 présente l'exigence  $E_2$  écrite en XACML.

Après avoir modélisé les deux exigences, on se pose la question qui est de savoir quelle réponse le système donnera à la requête d'un sujet qui travaille dans les deux structures. La réponse doit venir du patient et il appartient à l'administrateur des politiques de lui poser la question. La réponse détermine le choix de l'algorithme



### 4.3. MODÉLISATION DU CAS D'ÉTUDE EN XACML

programme 4.11 – Modélisation de l'exigence 2 du scénario 5

```
<Rule RuleId="exigence2" Effect="Permit">
  <Target/>
  <Condition>
    <Apply FunctionId="function:any-of">
      <Function FunctionId="string-equal"/>
      <AttributeValue> hematologie </AttributeValue>
      <AttributeDesignator
        Category="access-subject"
        AttributeId="subject:role"/>
    </Apply>
  </Condition>
</Rule>
```

de combinaison des règles qui sera utilisé dans la politique. Si Simon veut que le professionnel qui travaille dans les deux structures soit interdit, alors on utilisera **deny-overrides**. Si par contre il veut qu'il soit autorisé, on utilisera **permit-overrides**. Nous avons supposé dans notre modélisation que Simon autorise les professionnels travaillant dans les deux structures à accéder à ses dossiers.

**Tests après modélisation :** lorsque Pierre Bertrand qui travaille à l'urgence et en hématologie, envoie une demande d'accès à la ressource *RA\_002* de Simon, la réponse est **Permit**.

Lorsque Alice Fertier qui travaille uniquement à l'urgence, envoie une requête d'accès à la même donnée *RA\_002*, la réponse est **Deny**.

Lorsque Simon Lebon qui travaille uniquement en hématologie, envoie une requête d'accès à la même donnée *RA\_002*, la réponse est **Permit**.

Lorsque Simon Nadia qui travaille uniquement en Psychiatrie, envoie une requête d'accès à la même donnée *RA\_002*, la réponse est **NotApplicable**.

#### 4.3.7 Scénario 6

*Jeremy est un patient qui voudrait autoriser l'accès juste à son médecin traitant. Alors, l'accès sera automatiquement interdit une fois que le médecin n'est plus son médecin traitant.*

Notion : condition

### Solution du scénario 6 :

**Les exigences du scénario 6 :** une seule exigence  $E_1$  qui est le scénario lui-même.

La solution au scénario 6, telle que illustrée par le programme 4.12 consiste à :

- Créer une politique en précisant dans sa cible, qu'elle s'applique aux données de Jeremy.
- Créer une règle pour modéliser l'exigence  $E_1$ . La règle a pour effet **Permit** et possède une condition qui précise que tout sujet pouvant accéder aux ressources de Jeremy doit avoir le rôle de médecin traitant de Jeremy.

programme 4.12 – Modélisation du scénario 6

```

<Rule RuleId="exigence1" Effect="Permit">
  <Target/>
  <Condition>
    <Apply FunctionId="any-of">
      <Function FunctionId="string-equal"/>
      <AttributeValue > medecinTraitant </AttributeValue>
      <AttributeDesignator
        Category="access-subject"
        AttributeId="role"/>
    </Apply>
  </Condition>
</Rule>

```

Dans la table patients, il y a un attribut (*code\_professionnel*) qui permet de connaître le médecin traitant de chaque patient. Lorsqu'un sujet demande à accéder à une ressource, à travers l'identifiant de la ressource qui est demandée, on retrouve l'identifiant du patient qui est propriétaire de la ressource. Avec l'identifiant du sujet qui se trouve dans la requête et l'identifiant du patient qu'on a déterminé, on est à mesure maintenant à travers une requête dans la base de données de savoir si le sujet est oui ou non médecin traitant du patient. La chaîne de caractères *medecinTraitant* est ajoutée à la liste des rôles du sujet dès lors qu'on trouve que le sujet est bien le médecin traitant du patient.

### 4.3. MODÉLISATION DU CAS D'ÉTUDE EN XACML

Dès que Jeremy change de médecin traitant, le changement sera fait dans la base de données, mais il n'y a pas de changement à faire au niveau de la politique.

**Tests après modélisation :** Lorsque Pierre Bertrand, qui est le médecin traitant de Jeremy envoie une demande d'accès à la ressource *RA\_003* de Jeremy, la réponse est **Permit**.

Lorsque Simon Lebon, qui n'est pas le médecin traitant de Jeremy envoie une demande d'accès à la ressource *RA\_003* de Jeremy, la réponse est **NotApplicable**.

#### 4.3.8 Scénario 7

*Alice est une patiente qui voudrait autoriser l'accès aux personnels du service de la psychiatrie pour une durée de trois jours.*

**Solution du scénario 7 :**

**Les exigences du scénario 7 :** une seule exigence  $E_1$  qui est le scénario 7 lui-même.

La solution au scénario 7 consiste à :

- Créer une politique en précisant dans sa cible qu'elle s'applique aux données d'Alice.
- Créer une règle pour modéliser l'exigence  $E_1$ . La règle a pour effet **Permit**. Elle a une cible qui précise que le sujet doit être un personnel du service de santé mentale. La règle possède également une condition qui précise que la date d'émission de la requête, doit être inférieure à une date fixée en tenant compte des trois jours indiqués par le patient.

La représentation en XACML de l'exigence  $E_1$  est offerte dans le programme [4.13](#).

L'attribut d'environnement «*current-date*» indique la date à laquelle le sujet envoie sa requête. La valeur de l'attribut est contenue dans la requête. Cette date doit être inférieure à 2014-10-04.

programme 4.13 – Modélisation du scénario 7

```
<Rule RuleId="rule1" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue> psychiatrie</AttributeValue>
          <AttributeDesignator
            Category="access-subject"
            AttributeId="subject:role"/>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
  <Condition>
    <Apply FunctionId="date-less-than">
      <Apply FunctionId="date-one-and-only">
        <AttributeDesignator
          AttributeId="environment:current-date"
          Category="environment"/>
      </Apply>
      <AttributeValue>2014-10-04</AttributeValue>
    </Apply>
  </Condition>
</Rule>
```

**Tests après modélisation :** Lorsque Simone Bourger, une infirmière qui travaille à la Psychiatrie, demande le 2014-10-02 à accéder à la ressource *PSY\_002* d'Alice, la réponse est **Permit**.

Lorsque Alice Fertier, une infirmière qui travaille en urgence, demande le 2014-10-02 à accéder à la ressource *PSY\_002* d'Alice, la réponse est **NotApplicable**.

Lorsque Simone Bourger, une infirmière qui travaille à la Psychiatrie, demande le 2014-10-05 à accéder à la ressource *PSY\_002* d'Alice, la réponse est **NotApplicable**.

## 4.4 Critiques XACML

L'application de XACML au cas d'étude a révélé des limites de ce modèle. Ces limites se présentent comme suit :

1. L'administration des règles en XACML est très difficile. XACML est un langage

#### 4.4. CRITIQUES XACML

très complexe qu'il faut bien maîtriser avant de l'utiliser. Il existe des éditeurs de politiques XACML, mais leur utilisation requiert une bonne maîtrise de la norme XACML. Les règles sont très difficiles à lire et il est difficile d'investiguer pour connaître les permissions d'un utilisateur, et les utilisateurs qui ont accès à un objet donné.

2. Les systèmes se basant sur la priorité des règles, pour rendre la décision lors de l'évaluation des requêtes, sont difficilement modélisables en XACML. XACML offre des algorithmes de combinaison des règles et des politiques qui permettent de sélectionner une règle prioritaire pour rendre la décision. Cependant, avant d'utiliser ces algorithmes de combinaison, il faut d'abord parvenir à ordonner les règles suivant la priorité accordée à chaque règle. Les documents XACML étant des fichiers XML, il faut ordonner manuellement les règles. Lorsque les règles sont nombreuses, comme dans notre cas d'étude où les règles sont estimées en des centaines de milliers, il serait difficile de pouvoir faire cet ordonnancement manuel sans erreur. Une règle mal ordonnée peut être à l'origine de la violation de la politique de sécurité.
3. Il n'y a pas de contrainte définie entre les cibles des règles, des politiques et des ensembles de politiques. Pourtant, il faut une contrainte sur les cibles de ces trois éléments, pour éviter que certaines règles définies ne se perdent, et ne soient jamais utilisées pour évaluer les requêtes. L'algorithme d'évaluation des requêtes de XACML ressemble au parcours d'un arbre. Les règles constituent les feuilles de l'arbre. Pour qu'une règle  $r$  soit utilisée pour rendre la décision pour une requête  $q$ , il faut que la requête  $q$  s'apparie avec toutes les cibles situées au dessus de la règle  $r$ . Même si la règle  $r$  est applicable à la requête  $q$ ,  $r$  ne sera jamais atteinte si dans la branche de  $r$ , il y a une cible d'une politique ou d'un ensemble de politiques qui ne s'apparie pas avec  $q$ . La cible est définie en XACML comme un ensemble de requêtes auquel s'applique une règle, une politique ou un ensemble de politiques. Partant de cette définition et considérant  $r$  une règle,  $p$  une politique,  $s$  un ensemble de politiques et leur cible respective  $Tr$ ,  $Tp$  et  $Ts$ ; si  $r$  est contenue dans  $p$  et  $p$  dans  $s$ , alors toute mise à jour dans  $r$  ou dans  $p$  ou dans  $s$ , doit respecter cet invariant :  $Tr \subseteq Tp \subseteq Ts$ .
4. Le traitement d'une requête par XACML est très lent lorsque le système contient

## CHAPITRE 4. APPLICATION DE XACML AU CAS D'ÉTUDE

un grand nombre de règles. Le temps de réponse augmente linéairement avec l'augmentation du nombre de règles. Les résultats des tests effectués pour déterminer les performances de XACML sont présentés au tableau [6.1.10](#).

Le langage XACML est très riche. Il offre d'énormes possibilités pour représenter les politiques de contrôle d'accès. Il requiert cependant une très bonne maîtrise pour être utilisé. L'étude a permis de dégager les limites de XACML qui se résument au fait que XACML ne permet pas de modéliser la priorité entre les règles, l'administration des politiques qui est difficile, le risque d'avoir des règles qui ne seront jamais utilisées pour évaluer les requêtes et la performance qui se dégrade au fur et à mesure de l'augmentation du nombre de règles (temps de calcul linéaire par rapport au nombre de règles).

# Chapitre 5

## Application de SGAC au cas d'étude

À côté des modèles standards de contrôle d'accès rencontrés dans la littérature, il existe d'autres modèles que l'on pourrait appeler des modèles privés, développés par des entreprises pour répondre à leurs besoins. C'est le cas de la solution de gestion des accès et du consentement (SGAC), un modèle qui a été proposé pour implémenter le contrôle d'accès dans le réseau de santé de Sherbrooke. Le chapitre est subdivisé comme suit : la présentation du modèle SGAC, l'application du modèle SGAC à notre cas d'étude, les critiques du modèle SGAC.

### 5.1 Présentation du modèle SGAC

Les objectifs du modèle SGAC sont :

- contrôler les accès aux dossiers médicaux des patients,
- préserver la vie privée des patients dans un contexte de partage d'informations entre plusieurs structures sanitaires,
- prendre en compte le consentement du patient pour contrôler l'accès à ses données,
- s'assurer que les règles du contrôle d'accès sont conformes aux lois et réglementations en vigueur.

Pour atteindre ces objectifs, SGAC utilise un graphe des sujets, un graphe des ressources et des règles pour évaluer les requêtes des utilisateurs. Ces graphes sont orientés et acycliques. Les sous-sections qui suivent présentent l'expression des règles de contrôle d'accès en SGAC et le processus d'évaluation des requêtes.

### 5.1.1 Expression des règles en SGAC

Une règle est une directive qui permet de contrôler l'accès à une ressource. Une règle comprend :

- un *sujet* : une personne ou un groupe de personnes à contrôler ;
- une *ressource* : une donnée ou un groupe de données à protéger ;
- une *priorité* : un entier qui définit la priorité de la règle ;
- une *modalité* : une autorisation ou une interdiction ;
- une *condition* : une formule qui définit l'applicabilité de la règle.

Une requête en SGAC a le format suivant :

- *sujet* : l'initiateur de la requête
- *ressources* : un ensemble des ressources.
- *NIU* : l'identifiant du patient dans l'ensemble des hôpitaux de la région de Sherbrooke.

Par exemple : les figures 5.1 et 5.2 représentent respectivement le graphe des sujets et le graphe des ressources.

La règle utilisée pour cet exemple est : *Simon est un patient qui souhaite interdire l'accès à ses résultats de laboratoire aux infirmiers.*

Pour représenter la règle de Simon, le graphe des ressources est instancié (voir la figure 5.3) avec les paramètres qui définissent les ressources de Simon.

La règle de Simon s'écrit en SGAC comme suit :

*[Ressource= laboratoire (patient= Simon), sujet= infirmier ; modalité=interdiction ; priorité=2].*

### 5.1.2 Évaluation des requêtes

Il y a deux valeurs que le programme peut retourner après l'évaluation d'une requête. Les deux valeurs sont celles que l'attribut modalité peut prendre : *permission*



### 5.1. PRÉSENTATION DU MODÈLE SGAC

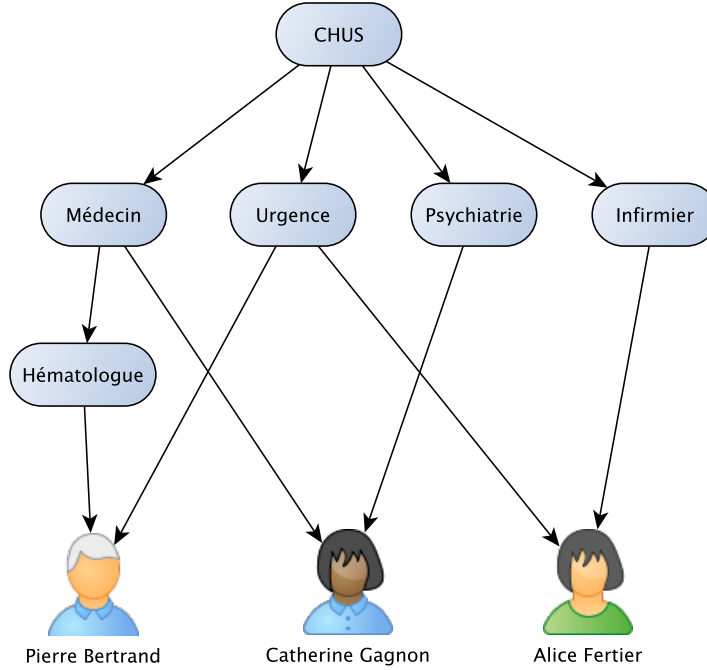


figure 5.1 – Hiérarchie des sujets

et *interdiction*. Le programme retourne la modalité de la règle qui a été sélectionnée pour évaluer la requête. Si aucune règle n'est applicable à la requête, le programme retourne *interdiction*. Le modèle SGAC évalue les requêtes en appliquant les deux formules ci-dessous. Dans les formules,  $p$  = priorité,  $s$  = sujet,  $r$  = ressource,  $n$  = identifiant unique du patient,  $c$  = condition,  $m$  = modalité.

**Formule 1 :** une règle  $L=(p,s_1,r_1(n_1),c,m)$  s'applique à une requête  $Q=(n_2,s_2,r_2)$  si et seulement si  $s_1 \geq s_2$  et  $r_1 \geq r_2$  et  $n_1 = n_2$  et  $c$  est satisfaite, où  $x \geq y$  signifie que  $x$  est un ancêtre de  $y$  dans le graphe (des sujets ou des ressources).

**Formule 2 :** si plusieurs règles sont applicables, alors elles sont ordonnées et c'est la règle la plus prioritaire qui est sélectionnée pour rendre la décision. Les règles applicables sont ordonnées comme suit :

soient  $L_1=(p_1,s_1,r_1(n_1),c_1,m_1)$  et  $L_2=(p_2,s_2,r_2(n_2),c_2,m_2)$  deux règles applicables,  $L_1 < L_2$  (i.e,  $L_1$  plus prioritaire que  $L_2$ ) si et seulement si

$$p_1 < p_2$$

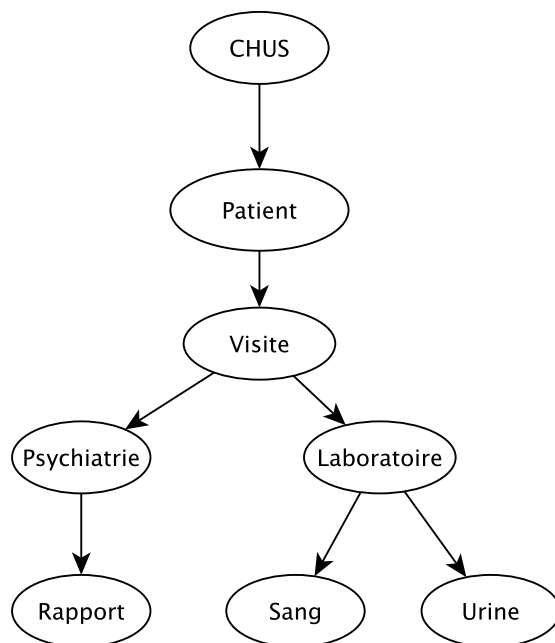


figure 5.2 – Graphe des types de ressources

ou

$p_1 = p_2$  et  $s_1 < s_2$ .

Si les deux règles ne sont pas comparables, alors l'interdiction l'emporte sur la permission.

L'algorithme d'implémentation de ces formules, est décrit ci-dessous (étapes 1 à 6). La requête suivante est utilisée pour cette description : *l'infirmière Alice Fertier souhaite accéder aux résultats de laboratoire de Simon.*

**Etape 1 :** le programme détermine les ancêtres du sujet *Alice Fertier* et les ancêtres de la ressource *laboratoire*. Au chargement des graphes des sujets et des ressources, le programme applique une fermeture transitive sur les graphes afin d'avoir les ancêtres du sujet et de la ressource. Les ancêtres du sujet *Alice Fertier* sont : *Infirmier*, *urgences*, *CHUS*. Les ancêtres de la ressource *laboratoire* sont : *Visite*, *Patient*, *CHUS*.

**Etape 2 :** le programme détermine les règles applicables au sujet, à la ressource et à

## 5.1. PRÉSENTATION DU MODÈLE SGAC

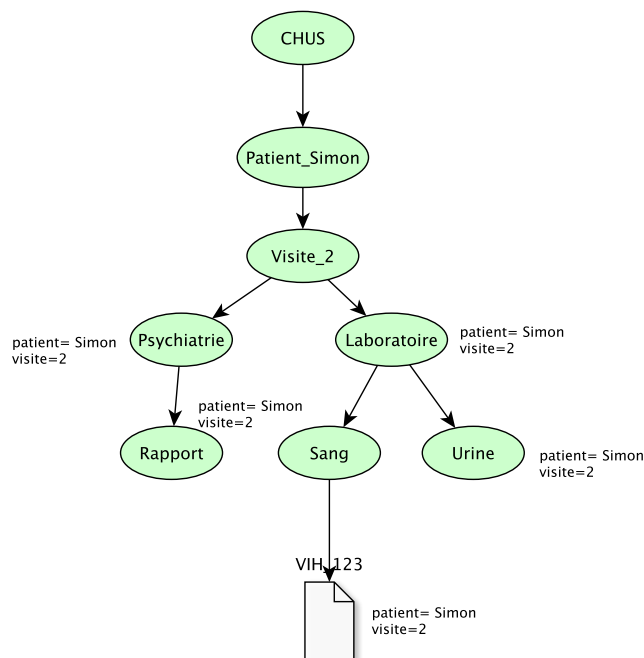


figure 5.3 – Graphe des types de ressources instancié avec les ressources de Simon

leurs ancêtres. Une fois les ancêtres du sujet et de la ressource obtenus, SGAC explore l'ensemble des règles pour garder les règles suivantes :

- l'ensemble des règles portant sur les ancêtres de *Alice Fertier* et sur *Alice Fertier*. Soit  $A$  cet ensemble.
- l'ensemble des règles portant sur les ancêtres de *laboratoire* et sur *laboratoire*. Soit  $B$  cet ensemble.

Si au moins un des ensembles  $A$  ou  $B$  est vide alors le programme retourne *interdiction* et l'évaluation est finie ; sinon l'évaluation se poursuit avec l'étape 3.

**Etape 3 :** le programme établit l'intersection entre les ensembles  $A$  et  $B$ . Soit  $T_1$  l'intersection entre  $A$  et  $B$ . Si  $T_1$  est vide alors le programme retourne *interdiction* et l'évaluation est finie ; sinon l'évaluation se poursuit avec l'étape 4.

**Etape 4 :** le programme détermine les règles applicables à la requête. Dans  $T_1$ , sont retirées les règles dont les paramètres ne correspondent pas à ceux de la requête. Par exemple, les règles qui portent sur des laboratoires d'un patient qui n'est pas Simon. Sont aussi retirées de  $T_1$  les règles qui ont des conditions évaluées à

*FAUX*. On obtient après toutes ces épurations un ensemble  $T_2$  qui est l'ensemble des règles applicables à la requête. Si  $T_2$  est vide alors le programme retourne *interdiction* et l'évaluation est finie. Sinon, si  $T_2$  contient une seule règle, alors le programme retourne la modalité de la seule règle et l'évaluation est finie. Sinon si  $T_2$  contient plusieurs règles et que l'ensemble de ces règles ont la même modalité, alors la modalité commune est retournée et l'évaluation est finie. Sinon si  $T_2$  contient plusieurs règles ayant des modalités différentes, on passe à l'étape 5 et les suivantes qui font la gestion des conflits.

**Etape 5 :** le programme trie des règles de  $T_2$  par priorité. Le tri par priorité est trivial car les priorités sont des valeurs numériques. Si après le tri des règles de  $T_2$ , il y a une seule règle qui a une priorité supérieure au reste des règles, cette règle est sélectionnée, le programme retourne sa modalité et l'évaluation est finie. Sinon, soit  $T_3$  l'ensemble des règles prioritaires. Si l'ensemble des règles de  $T_3$  ont la même modalité, alors le programme retourne la modalité commune et l'évaluation est finie. Sinon (l'ensemble des règles de  $T_3$  ont la même priorité et des modalités différentes), suit alors l'étape 6.

**Etape 6 :** le programme trie les règles de  $T_3$  par sujets. La hiérarchie entre les sujets est une relation d'ordre partiel strict. Ce qui permet, lorsque deux sujets sont comparables, de déterminer lequel est inférieur à l'autre (lequel est plus profond que l'autre en considérant le graphe du sujets). Deux sujets sont comparables si l'un est l'ancêtre de l'autre par fermeture transitive du graphe des sujets. Deux sujets sont incomparables si on ne peut pas établir un lien entre eux par fermeture transitive. Si deux sujets  $s_1$  et  $s_2$  sont comparables et  $s_1$  est l'ancêtre de  $s_2$  alors  $s_2 < s_1$ . On dit aussi que  $s_2$  est plus spécifique que  $s_1$ . Si tous les sujets des règles de  $T_3$  sont comparables, alors  $T_3$  est trié par ordre croissant des sujets. La règle ayant le plus petit sujet est sélectionnée, le programme retourne sa modalité et l'évaluation est finie. Sinon (les sujets dans  $T_3$  ne sont pas comparables), le programme retourne *interdiction* et l'évaluation est finie.

### 5.2 Modélisation du cas d'étude

Suivant la description du modèle présentée plus haut, deux graphes décrivant respectivement la hiérarchie des sujets et la hiérarchie des ressources doivent être construits avant de commencer la modélisation des règles. Le graphe des sujets de la figure 2.2 et le graphe des ressources de la figure 2.3 sont utilisés pour modéliser le cas d'étude. Les sous-sections 5.2.1 à 5.2.7 présentent la modélisation de chaque scénario du cas d'étude.

#### 5.2.1 Scénario 1

*Patrice est un patient qui souhaite interdire l'accès à l'ensemble de ses données personnelles (dépistage du VIH, test ADN, Examens psychiatriques) à l'ensemble des professionnels de l'hôpital.*

Notion : Traitement d'une seule règle

#### Solution du scénario 1

##### Les exigences du scénario 1

$E_{1.1}$  : identique au scénario 1.

En instanciant le graphe des ressources avec les données de Patrice, on obtient le résultat présenté à la figure 5.4. Les paramètres définissant les données de Patrice sont affectés aux noeuds du graphe des ressources lors de l'instanciation.

La modélisation de la règle de Patrice consiste à interdire l'accès aux noeuds descendants du noeud *patient* dans le graphe des ressources figure 5.4. Le noeud *patient* prend la valeur de l'identifiant unique du patient Patrice. Le noeud *hôpital* sur le graphe de sujets (figure 2.2) regroupe l'ensemble des sujets de l'hôpital. Toute règle qui s'applique au noeud *hôpital*, s'applique à tous les sujets de l'hôpital. Le tableau 5.1 présente la modélisation du scénario 1 dans le modèle SGAC. La priorité 2 est utilisée pour toutes les règles émanant d'un patient. Il n'y a aucune condition dans cette règle, la valeur *VRAI* est alors affectée à l'attribut condition.

## CHAPITRE 5. APPLICATION DE SGAC AU CAS D'ÉTUDE

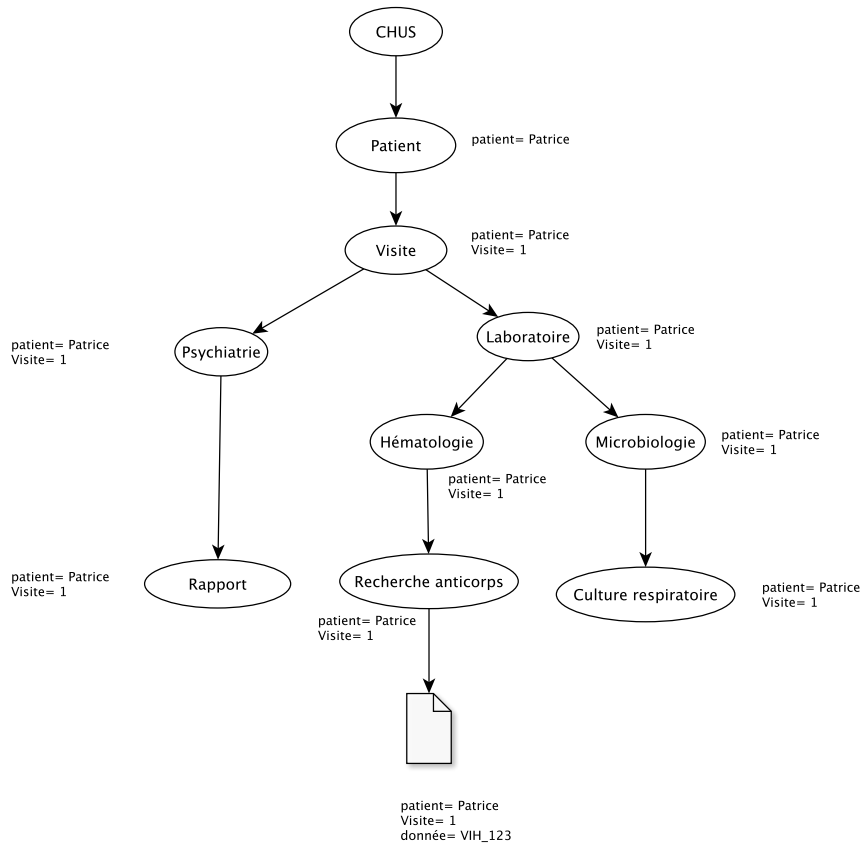


figure 5.4 – Graphe des types de ressources instancié avec les ressources du patient Patrice

N	règle	Ressource	Sujet	Priorité	Modalité	Condition
1	$E_{1.1}$	patient=Patrice	CHUS	2	interdiction	true

tableau 5.1 – Modélisation du scénario 1

Supposons que Pierre Bertrand veuille accéder aux laboratoires de Patrice. La requête suivante est envoyée au moteur de contrôle d'accès :  $\langle \text{sujet} = \text{Pierre Bertrand}, \text{ressource} = \text{laboratoire} (\text{patient} = \text{Patrice}) \rangle$

A la réception de la requête, le programme détermine par fermeture transitive des graphes inversés, les ancêtres du sujet Pierre Bertrand et de la ressource *laboratoire*. Le noeud *hôpital* fait partie des ancêtres du sujet Pierre Bertrand et le noeud *patient* fait partie des ancêtres de la ressource *laboratoire*. Le programme détermine ensuite

## 5.2. MODÉLISATION DU CAS D'ÉTUDE

les règles applicables à la requête de Pierre Bertrand. Les règles applicables sont celles qui portent à la fois sur le sujet de la requête (ou sur un ancêtre du sujet), et sur la ressource de la requête (ou sur un ancêtre de la ressource). La règle de Patrice est bien applicable à la requête de Pierre Bertrand. En supposant que c'est la seule règle applicable à cette requête, la réponse retournée = *"interdiction"* qui correspond à la modalité de la règle de Patrice.

### 5.2.2 Scénario 2

*Alain est un patient qui souhaite autoriser l'accès à l'ensemble de ses données personnelles (dépistage du VIH, test ADN, Examens psychiatriques) à l'ensemble des professionnels de l'hôpital. Cela dit, il y a une loi qui précède la règle d'Alain et qui interdit l'accès aux examens psychiatriques de tous les patients.*

Notion : Gestion de la priorité entre deux règles.

#### Solution du scénario 2

##### Les exigences du scénario 2

$E_{2.1}$  : *la loi interdit l'accès aux examens psychiatriques de tous les patients.*

$E_{2.2}$  : *Alain souhaite autoriser l'accès à l'ensemble de ses données personnelles (dépistage du VIH, test ADN, Examens psychiatriques) à l'ensemble des professionnels de l'hôpital.*

$E_{2.3}$  : *la loi est prioritaire par rapport aux règles du patient*

Pour la modélisation du scénario 2, le graphe des ressources est instancié avec les données de Alain. Les exigences  $E_{2.1}$  et  $E_{2.2}$  correspondent à des règles et sont modélisées séparément. La priorité entre les règles ( $E_{2.3}$ ) est modélisée en affectant des valeurs numériques à l'attribut *priorité*. Une règle  $r_1$  est prioritaire par rapport à une règle  $r_2$  si la valeur de la priorité de  $r_1$  est inférieure à celle de  $r_2$ . La valeur 1 est affectée aux règles émanant de la loi et la valeur 2 est affectée aux règles émanant du patient. Le tableau 5.2 présente la modélisation du scénario 2 dans le modèle SGAC.

## CHAPITRE 5. APPLICATION DE SGAC AU CAS D'ÉTUDE

N	règle	Ressource	Sujet	Priorité	Modalité	Condition
2	$E_{2.1}$	psychiatrie	CHUS	1	interdiction	true
2	$E_{2.2}$	patient= Alain	CHUS	2	permission	true

tableau 5.2 – Modélisation du scénario 2

Supposons que Pierre Bertrand veuille accéder aux examens psychiatriques de Alain. La requête suivante est envoyée au moteur de contrôle d'accès :  $\langle \text{ sujet= Pierre Bertrand, ressource= examens psychiatriques (patient= Alain)} \rangle$ .

Pierre Bertrand fait partie du groupe de sujets *hôpital*. Aussi, les noeuds *psychiatrique* et *patient* sont des ancêtres de la ressource de la requête. Les deux règles  $E_{2.1}$  et  $E_{2.2}$  sont applicables à la requête de Pierre Bertrand. A cause de la priorité de la règle  $E_{2.1}$  qui est inférieure à celle de  $E_{2.2}$ , alors la réponse à la requête = "*interdiction*".

### 5.2.3 Scénario 3

*Romain est un patient qui souhaite interdire l'accès aux laboratoires à l'infirmière Alice Fertier. Cela dit, Romain avait déjà une directive qui autorisait toutes les infirmières à accéder à ses laboratoires.*

Notion : Hiérarchie des sujets.

### Solution du scénario 3

#### Les exigences du scénario 3

$E_{3.1}$  : *Romain souhaite interdire l'accès à ses résultats de laboratoires à l'infirmière Alice Fertier.*

$E_{3.2}$  : *Romain avait déjà une directive qui autorisait toutes les infirmières à accéder à ses laboratoires.*

Chacune des exigences  $E_{3.1}$  et  $E_{3.2}$  constitue une règle.

Le tableau 5.3 présente la modélisation du scénario 3 dans le modèle SGAC.

Supposons que Alice Fertier veuille accéder aux résultats de laboratoires de Romain. La requête suivante est envoyée au moteur de contrôle d'accès :  $\langle \text{ sujet= Alice Fertier, ressource= laboratoires (patient= Romain)} \rangle$  Les deux règles  $E_{3.1}$  et  $E_{3.2}$  sont



## 5.2. MODÉLISATION DU CAS D'ÉTUDE

N	règle	Ressource	Sujet	Priorité	Modalité	Condition
3	$E_{3.1}$	laboratoire (patient= Romain)	Alice Fertier	2	interdiction	true
3	$E_{3.2}$	laboratoire (patient= Romain)	infirmière	2	permission	true

tableau 5.3 – Modélisation du scénario 3

applicables à la requête de Alice Fertier. Les deux règles ont la même priorité. Le modèle SGAC propose en cas d'égalité des priorités, d'ordonner les règles sur la base de la hiérarchie des sujets. Le sujet *infirmière* fait partie des ancêtres du sujet *Alice Fertier*, donc  $Alice\ Fertier < infirmière$ . Selon la spécification de SGAC, en ayant  $Alice\ Fertier < infirmière$ , la règle portant sur *Alice Fertier* a préséance sur celle portant sur *infirmière*. La règle  $E_{3.1}$  est choisie pour évaluer la requête et par conséquent, la réponse à la requête de Alice Fertier = "*interdiction*".

### 5.2.4 Scénario 4

*Romain est un patient qui possède dans son dossier deux examens de laboratoire : laboratoire1 et laboratoire2. Il spécifie une règle, autorisant les médecins à accéder à ses laboratoires. Romain ajoute ensuite à son dossier un autre laboratoire : laboratoire3. Le Dr. Bernard Lapointe lance une requête pour accéder au laboratoire3 de Romain.*

Notion : Nouvelles données.

**Solution du scénario 4 :**

**Les exigences du scénario 4 :**

$E_{4.1}$  : *Romain autorise les médecins à toujours accéder à ses résultats de laboratoires y compris les nouveaux laboratoires.*

Dans  $E_1$ , *ressource = laboratoires(patient = Romain), sujet = médecin*

Le tableau 5.4 présente la modélisation du scénario 4 dans le modèle SGAC.

Tous les laboratoires de Romain (actuels et futurs), ont pour ancêtre le noeud *laboratoire*. En basant la règle  $E_{4.1}$  de Romain sur le noeud *laboratoire*, elle s'applique

## CHAPITRE 5. APPLICATION DE SGAC AU CAS D'ÉTUDE

N	règle	Ressource	Sujet	Priorité	Modalité	Condition
4	$E_{4.1}$	laboratoire (patient=Romain)	médecin	2	permission	true

tableau 5.4 – Modélisation du scénario 4

aussi bien aux laboratoires actuels et futurs de Romain. Les médecins ont de ce fait accès aux futurs laboratoires de Romain.

### 5.2.5 Scénario 5

*Simon est un patient qui a spécifié deux règles, la première interdit aux professionnels travaillant à l'urgence d'accéder à son dossier et la seconde autorise les professionnels travaillant en hématologie d'accéder au dossier. Dr. Pierre Bertrand souhaite accéder au dossier de Simon.*

Notion : Multi-appartenance

**Solution du scénario 5 :**

**Les exigences du scénario 5 :**

$E_{5.1}$  : *Simon interdit d'abord les professionnels travaillant à l'urgence d'accéder à son dossier*

$E_{5.2}$  : *Simon autorise ensuite les professionnels travaillant en hématologie d'accéder au dossier*

Chacune des exigences  $E_{5.1}$  et  $E_{5.2}$  constitue une règle.

Pour  $E_{5.1}$ , *ressource = patient Simon, sujet = urgence*

Pour  $E_{5.2}$ , *ressource = patient Simon, sujet = hématologie*

Le tableau 5.5 présente la modélisation du scénario 5 dans le modèle SGAC.

Dans le graphe des sujets, Pierre Bertrand travaille à la fois en *urgence* et en *hématologie*. Lorsque Pierre Bertrand demande à accéder au dossier de Simon, les deux règles  $E_{5.1}$  et  $E_{5.2}$  sont applicables à sa requête. Les deux règles ont la même priorité et les sujets *urgence* et en *hématologie* sont incomparables. Selon SGAC, c'est

## 5.2. MODÉLISATION DU CAS D'ÉTUDE

N	règle	Ressource	Sujet	Priorité	Modalité	Condition
5	$E_{5.1}$	patient= Simon	urgentologue	2	interdiction	true
5	$E_{5.2}$	patient= Simon	hématologue	2	permission	true

tableau 5.5 – Modélisation du scénario 5

la règle dont la modalité est *interdiction* qui est sélectionnée pour rendre la décision. Pierre Bertrand ne peut donc pas accéder au dossier de Simon.

### 5.2.6 Scénario 6

*Jeremy est un patient qui voudrait autoriser l'accès juste à son médecin traitant. Alors, l'accès sera automatiquement interdit une fois le médecin n'est plus son médecin traitant.*

Notion : condition

**Solution du scénario 6 :**

**Les exigences du scénario 6 :**  $E_{6.1}$  : idem scénario 6.

Le médecin traitant de Jeremy n'est pas présent sur le graphe des sujets. Pour modéliser la règle de Jérémy, l'attribut *condition* de la règle est utilisé pour définir un prédicat dont l'évaluation vérifie si le sujet de la requête est le médecin traitant de Jérémy. Le tableau 5.6 présente la modélisation du scénario 6 dans le modèle SGAC.

N	règle	Ressource	Sujet	Priorité	Modalité	Condition
7	$E_{7.1}$	patient=Jeremy	CHUS	2	permission	sujet requête= médecin traitant de Jeremy

tableau 5.6 – Modélisation du scénario 6

### 5.2.7 Scénario 7

*Alice est une patiente qui voudrait autoriser l'accès aux personnels du service de psychiatrie pour une durée de 3 jours.*

Notion : Contexte temps

**Solution du scénario 7 :**

**Les exigences du scénario 7 :** il y a une seule exigence  $E_{7.1}$  qui est identique au scénario 7.

La condition sur le temps de  $E_{7.1}$  est modélisée à travers l'attribut *condition*. La durée de 3 jours permet de calculer la date de fin de l'autorisation. Les requêtes doivent être envoyées à des dates inférieures ou égales à la date de fin de l'autorisation pour avoir la permission d'accéder aux dossiers de Alice.

Le tableau 5.7 présente la modélisation du scénario 7 dans le modèle SGAC.

N	règle	Ressource	Sujet	Priorité	Modalité	Condition
8	$E_{7.1}$	patient=Alice	psychiatre	2	permission	date requête $\leq$ date fixée

tableau 5.7 – Modélisation du scénario 7

Le tableau 5.8 présente le résumé de la modélisation de l'ensemble des scénarios du cas d'étude.

N	règle	Ressource	Sujet	Priorité	Modalité	Condition
1	$E_{1.1}$	patient=Patrice	CHUS	2	interdiction	true
2	$E_{2.1}$	psychiatrie	CHUS	1	interdiction	true
2	$E_{2.2}$	patient= Alain	CHUS	2	permission	true
3	$E_{3.1}$	laboratoire (patient= Romain)	Alice Fertier	2	interdiction	true
3	$E_{3.2}$	laboratoire (patient= Romain)	infirmière	2	permission	true
4	$E_{4.1}$	laboratoire (patient=Romain)	médecin	2	permission	true
5	$E_{5.1}$	patient= Simon	urgentologue	2	interdiction	true
5	$E_{5.2}$	patient= Simon	hématologue	2	permission	true
6	$E_{6.1}$	patient=Jeremy	CHUS	2	permission	sujet requête= médecin traitant de Jeremy
7	$E_{7.1}$	patient=Alice	psychiatre	2	permission	date requête $\leq$ date fixée

tableau 5.8 – Résumé de la modélisation de l'ensemble des scénarios

### 5.3. CRITIQUES DE SGAC

## 5.3 Critiques de SGAC

L'étude de SGAC a révélé les problèmes suivants :

**Critique 1 :** la modalité *interdiction* retournée lors de l'évaluation d'une requête n'est pas explicite. On ne sait pas si l'interdiction est due à une règle d'interdiction, à un problème interne du système, ou à l'absence de règle applicable à la requête. Le modèle SGAC pourrait s'inspirer des quatre valeurs que retourne XACML (*permis, interdit, non applicable et indéterminé*) qui sont plus explicites aux yeux de l'utilisateur.

**Critique 2 :** les attributs suivants manquent dans le modèle SGAC :

**action :** le modèle SGAC contrôle une action qui n'est pas connue. Aucun attribut dans le langage du modèle ne permet de spécifier l'action qui est contrôlée. Le modèle n'est pas facilement extensible si on veut préciser les actions à contrôler. Les utilisateurs peuvent *lire, créer, modifier ou supprimer* les données. Dans ces conditions, on peut vouloir définir des règles pour des actions particulières. Par exemple, autoriser les infirmiers à lire seulement les données et autoriser les médecins à lire et modifier les données. Le manque de l'attribut *action* fait qu'on ne peut pas préciser les actions dans les règles.

**obligation :** les obligations servent à spécifier des actions qui seront exécutées conjointement avec l'application de la décision rendue après évaluation d'une requête. Par exemple envoyer un courriel au patient quand un utilisateur accède à son dossier.

Le modèle SGAC se caractérise par un langage simple à comprendre. Il permet de décrire facilement les règles de contrôle d'accès. Il offre également une grande facilité d'administration du système. Cependant SGAC manque d'attributs pour spécifier les actions et les obligations.

## CHAPITRE 5. APPLICATION DE SGAC AU CAS D'ÉTUDE

## Chapitre 6

# Identification des critères d'évaluation et comparaison de RBAC, XACML et SGAC

Les études menées dans les chapitres précédents permettent d'avoir une bonne connaissance des modèles de contrôle d'accès. Les applications de RBAC, XACML et SGAC au cas d'étude, ont permis d'appréhender les avantages et les inconvénients de chacun de ces modèles. Dans ce chapitre, nous définissons un cadre pour comparer les modèles de contrôle d'accès dans le domaine des dossiers médicaux électroniques. Le chapitre comprend l'identification des critères d'évaluation et la comparaison des modèles RBAC, XACML et SGAC.

### 6.1 Critères d'évaluation

L'une des spécificités des mécanismes de protection de la vie privée des personnes, est la conformité de ces mécanismes aux lois et aux règlements. L'évaluation des modèles devant servir à implémenter ces mécanismes, ne peut se faire sans tenir compte des lois et des règlements. De nombreux textes gouvernementaux ou émanant d'organismes œuvrant dans le domaine de la protection de la vie privée des personnes, sont définis pour encadrer la collecte et l'utilisation des données à caractère personnel.

## CHAPITRE 6. IDENTIFICATION DES CRITÈRES D'ÉVALUATION ET COMPARAISON DE RBAC, XACML ET SGAC

*L'Organisation de coopération et de développement économique (OCDE)* a publié en 1980 les lignes directrices[1] régissant la protection de la vie privée et les flux transfrontaliers de données à caractère personnel. La loi américaine HIPAA, adoptée en 1996, définit les normes pour assurer la disponibilité, l'intégrité et la confidentialité des données médicales des patients, enregistrées ou transmises par voies électroniques. Aussi en 1996, le Conseil canadien des normes a approuvé à titre de norme nationale, un ensemble de principes sur la protection des données à caractère personnel, mis au point par l'Association canadienne de normalisation (CSA : Canadian Standards Association). Les principes de l'OCDE et de la CSA ont été repris dans les lois fédérales du Canada sur la protection de la vie privée [8]. L'exploitation de ces textes sert à cerner les exigences du contrôle d'accès dans le domaine de la santé. Nous nous sommes également appuyés sur les exigences [13] identifiées par l'organisme canadien *Inforoute Santé Canada*. Les principales exigences du contrôle d'accès aux dossiers médicaux électroniques se résument aux points suivants :

1. la conformité avec les lois et les règlements ;
2. la prise en compte du consentement du patient ;
3. la conformité des accès aux buts (exemples de buts : raison de soins, recherches scientifiques) ;
4. la facilité des audits ;
5. la facilité des investigations, pour connaître les permissions d'un utilisateur ;
6. la performance du système ;
7. la possibilité d'intégration avec d'autres systèmes ;
8. le respect du partage des tâches (les permissions sont généralement accordées selon les compétences des utilisateurs) ;
9. la prise en compte des contraintes temporelles et géographiques ;
10. le respect des obligations (exemple : obligation d'informer le patient quand on accède à son dossier) ;
11. la sûreté du système ;
12. la facilité d'administration ;



## 6.1. CRITÈRES D'ÉVALUATION

13. le pouvoir de délégation des permissions ;
14. la journalisation des accès.

Dans les sous-sections qui suivent, nous décrivons les critères d'évaluation des modèles de contrôle d'accès en regard des exigences du contrôle d'accès aux dossiers médicaux électroniques.

### 6.1.1 Expression des politiques

Cette propriété évalue la capacité du modèle à représenter toutes les règles de la politique de contrôle d'accès d'un système. RBAC n'est pas très expressif pour modéliser les scénarios du cas d'étude choisi. RBAC ne peut pas modéliser les interdictions, la hiérarchie des ressources et les règles dépendant d'un contexte. XACML permet de modéliser les scénarios du cas d'étude. Cependant, il est difficile d'ordonner les règles suivant leur priorité. SGAC permet de modéliser les scénarios du cas d'étude. Cependant, SGAC ne permet pas de spécifier les opérations dans les règles.

### 6.1.2 Administration des politiques

Au regard du nombre élevé des règles contenues dans la politique de contrôle d'accès, le modèle doit offrir de grandes facilités de gestion de ces règles. Les modèles sont évalués à travers le nombre d'opérations à effectuer pour créer une règle, à travers le nombre d'opérations à effectuer pour assigner et révoquer les permissions d'un utilisateur. Le modèle doit permettre de pouvoir facilement faire des investigations sur l'assignation des permissions aux utilisateurs. Par exemple, l'administrateur doit être en mesure de savoir l'ensemble des objets auxquels un utilisateur a accès, et inversement, l'ensemble des utilisateurs qui ont accès à un objet donné. A travers les applications des modèles au cas d'étude, il ressort que l'administration des politiques en RBAC n'est pas aisée. En effet, la hiérarchie des ressources n'étant pas prise en considération dans le modèle, cela entraîne la création des permissions sur chacune des données, même si la règle porte sur une même catégorie de données. Par exemple lorsqu'une règle porte sur les laboratoires d'un patient, il faut créer  $n$  permissions pour les  $n$  laboratoires du patient. Aussi, le besoin d'exprimer des règles plus granulaires,

entraîne l'explosion des rôles qui rend l'administration difficile. RBAC offre par compte une bonne lisibilité des règles et des fonctions qui permettent de savoir les utilisateurs qui ont accès à un objet donné, et les objets auxquels un utilisateur peut accéder. L'administration des règles est facile dans SGAC. Il n'y a pas de multiplication de permissions à créer, du fait de la prise en compte de la hiérarchie des ressources et des sujets. Lorsqu'une règle porte sur les laboratoires d'un patient, la règle est définie sur le nœud *laboratoire* et elle s'applique à tous les nœuds enfants de *laboratoire*. L'administrateur n'a donc pas besoin de créer la règle pour chacun des laboratoires du patient.

XACML offre des profils qui permettent de modéliser la hiérarchie des ressources et des rôles. Il n'y a pas de multiplication de permissions à créer lorsqu'il s'agit de modéliser une règle qui porte sur une catégorie de données. Cependant l'administration des règles en XACML est difficile. XACML est un langage très complexe mais qu'il faut bien maîtriser avant de l'utiliser. Il existe des éditeurs de politiques XACML [3], conçus pour faciliter la rédaction des politiques XACML. Cependant l'utilisation de ces éditeurs demandent également une bonne maîtrise de XACML. En plus, les règles sont très difficiles à lire, et il est difficile d'investiguer pour connaître les permissions d'un utilisateur.

### 6.1.3 Autoriser les exceptions (*Break the glass*)

Le «Break the glass»(BTG) est une technique qui vise à autoriser un praticien à voir tout ou une partie du dossier médical d'un patient en cas d'urgence, lorsque ce praticien n'a pas les privilèges nécessaires pour accéder audit dossier. Le BTG est recommandé dans *HIPAA* et dans les exigences de *Inforoute Santé Canada*. La pratique dans certaines organisations pour intégrer une politique BTG, consiste à créer un compte temporaire avec des privilèges très élevés, et affecter ce compte à l'utilisateur qui a besoin d'accéder aux ressources [51]. Non seulement cette technique est inefficace car les permissions étant fonction du patient, il faudrait que l'administrateur des politiques soit toujours présent pour octroyer les privilèges exceptionnels. En plus, cette technique peut conduire à une violation de la politique de sécurité au cas où le compte créé tombe dans les mains d'une personne malveillante. Achim D. Brucker

## 6.1. CRITÈRES D'ÉVALUATION

et Helmut Petritsch [19] proposent d'intégrer directement le BTG dans le modèle de contrôle d'accès. Les modèles RBAC et XACML n'intègrent pas le BTG dans leur spécification. Ana Ferreira et al. [21, 5] étendent RBAC en ajoutant le BTG. Brucker and Helmut Petritsch [19] étendent XACML en ajoutant le BTG. SGAC gère le BTG à travers les priorités et les conditions définies dans les règles, et un mécanisme de remémoration du BTG. Les requêtes des sujets contiennent un attribut booléen qui se positionne à *VRAI* en cas d'urgence. Lorsque cet attribut est à *VRAI*, le sujet a accès à la ressource qu'il demande même s'il y a des règles qui l'interdisent.

### 6.1.4 Principe de moindre privilège

Le principe de moindre privilège stipule qu'un utilisateur doit avoir le minimum de privilèges qui lui permettent de remplir sa tâche au sein de l'organisation. L'application de ce principe limite les dommages qui peuvent résulter d'un accident, d'une erreur, ou d'utilisation non autorisée. Pour que le principe de moindre privilège soit respecté, il faut identifier la fonction de chaque sujet, déterminer l'ensemble minimum de privilèges nécessaires pour remplir chaque fonction et restreindre le sujet aux privilèges qui lui sont nécessaires. En donnant à un sujet, un privilège qui ne lui est pas nécessaire pour remplir sa fonction, il peut utiliser ce privilège pour contourner la politique de sécurité de l'organisation. RBAC permet de respecter le principe de moindre privilège. Pour appliquer le principe de moindre privilège avec RBAC, il suffit d'assigner aux rôles de chaque utilisateur les permissions nécessaires à l'accomplissement de sa fonction. L'application des contraintes de séparation des tâches permet de restreindre l'utilisateur aux seuls rôles nécessaires à l'accomplissement de sa fonction.

Il est difficile avec XACML, de respecter le principe de moindre privilège. En effet, XACML à travers le profil *core and hierarchy RBAC*, modélise une partie de RBAC. XACML ne modélise pas l'assignation des utilisateurs aux rôles et les contraintes de séparation des tâches, qui sont nécessaires pour appliquer le principe de moindre privilège.

Le principe de moindre privilège peut être appliqué avec SGAC. En SGAC, par définition, un sujet  $s$  a accès en temps normal (hormis les cas d'urgence) à une ressource  $r$ , si

- i. il existe une règle qui autorise  $s$  à accéder à  $r$  ou à un ancêtre de  $r$ , ou
- ii. il existe une règle qui autorise un ancêtre de  $s$  à accéder à  $r$  ou à un ancêtre de  $r$ .

De même, un sujet  $s$  est interdit d'accéder en temps normal (hormis les cas d'urgence) à une ressource  $r$ , si

- i. il existe une règle qui interdit à  $s$  d'accéder à  $r$  ou à un ancêtre de  $r$ , ou
- ii. il existe une règle qui interdit à un ancêtre de  $s$  d'accéder à  $r$  ou à un ancêtre de  $r$ .

Pour restreindre un sujet  $s$  à un ensemble  $Rs$  des ressources auxquelles  $s$  peut accéder, on peut définir une règle de forte priorité qui interdit à  $s$  l'accès à l'ensemble des ressources appartenant au complémentaire de  $Rs$ . La règle est définie sur  $s$  et non sur un ancêtre de  $s$ . Lorsqu'un ancêtre de  $s$  est autorisé à accéder à une ressource  $r$  telle que  $r \notin Rs$ ,  $s$  ne peut pas accéder à  $r$ , à cause de la priorité de la règle qui interdit  $s$ , et même en cas de priorité entre la règle qui interdit  $s$  et la règle qui l'autorise à travers son ancêtre, c'est la règle d'interdiction qui sera appliquée car le sujet  $s$  est plus spécifique que son ancêtre. il est alors possible de respecter le principe de moindre privilège en SGAC.

### 6.1.5 Délégation des privilèges

La délégation est un mécanisme qui permet à un utilisateur  $A$  d'agir au nom d'un autre utilisateur  $B$  en faisant en sorte que les droits d'accès de  $B$  soient disponibles pour  $A$ . Il est de coutume dans le domaine de la santé qu'un médecin avant d'aller en congé délègue ses permissions à un autre qui assurera le suivi de ses patients pendant son absence. Une opération de délégation change temporairement l'état du contrôle d'accès. En raison de son effet sur l'état du contrôle d'accès, la délégation peut conduire à une violation des politiques de sécurité, en particulier dans les séparations statiques des tâches [57]. Par exemple, si les rôles  $R_1$  et  $R_2$  sont mutuellement exclusifs, un utilisateur qui est un membre de  $R_1$  ne devrait pas être autorisé à recevoir  $R_2$  d'autres utilisateurs par délégation. La délégation est une fonctionnalité qui devrait figurer sur les modèles de contrôle d'accès. Aucun des modèles étudiés ne modélise

## 6.1. CRITÈRES D'ÉVALUATION

explicitement la délégation des privilèges. Beaucoup des travaux de recherches sont faits sur la délégation et la plupart de ces travaux portent sur le modèle RBAC [57].

### 6.1.6 Expression des conditions

Les conditions raffinent l'applicabilité des règles. Par exemple, certaines autorisations sont définies en fonction du temps, du lieu, de l'historique des accès, etc. Le modèle de contrôle d'accès doit permettre d'exprimer ces conditions. Les conditions ne sont pas modélisables en RBAC. Les modèles SGAC et XACML offrent la possibilité de définir des conditions sur les règles.

### 6.1.7 Contrainte de séparation des tâches

Dans le domaine de la santé, les autorisations sont également basées sur les compétences des utilisateurs. Par exemple, les infirmiers peuvent consulter les patients mais ne peuvent pas prescrire une ordonnance médicale. La séparation des tâches est un moyen qui permet de ne pas accorder à un utilisateur les permissions qui ne sont pas liées à ses compétences. Dans les modèles étudiés, seul RBAC prend en compte explicitement la séparation statique et dynamique des tâches.

### 6.1.8 Mécanisme de résolution de conflits

Il y a conflit lorsque, pendant l'évaluation d'une requête, le système se retrouve avec deux ou plusieurs règles applicables ayant des effets différents. Certaines règles autorisent l'accès pendant que d'autres l'interdisent. Le choix d'une seule règle applicable pour rendre la décision dépend du mécanisme implementé dans le système et prévu dans le modèle. Lorsqu'un modèle standard prévoit déjà un mécanisme de résolution des conflits, l'utilisateur doit analyser l'algorithme de ce mécanisme afin de s'assurer qu'il s'adapte ou est extensible pour s'adapter aux besoins de l'entreprise. XACML offre des algorithmes de combinaison qui permettent de résoudre les conflits entre les règles. SGAC intègre la gestion de conflits entre les règles. En RBAC, il n'y a pas de conflit tel que défini préalablement. Il peut avoir en RBAC, des conflits liés à la définition des contraintes. Par exemple, si nous avons deux contraintes  $C_1$  et  $C_2$

définies de la sorte :

- $C_1$  : les rôles  $R_1$  et  $R_2$  sont contradictoires en raison de la séparation des tâches, c'est à dire que les deux rôles ne peuvent pas être assignés à un même utilisateur ;
- $C_2$  :  $R_1$  est un préalable à  $R_2$  c'est à dire que  $R_2$  peut être assigné à un utilisateur seulement si  $R_1$  a été assigné à cet utilisateur.

$C_1$  et  $C_2$  sont incompatibles. RBAC ne propose pas un mécanisme de résolution de conflits.

### 6.1.9 Prise en compte des interdictions et des permissions

Dans une politique de contrôle d'accès, on retrouve des interdictions et des permissions. Lorsqu'un modèle de contrôle d'accès modélise une seule catégorie de règles, il y a possibilité de violer les règles de la catégorie non prise en compte par le modèle. RBAC permet de modéliser uniquement les permissions. XACML et SGAC offrent la possibilité de modéliser les deux catégories de règles.

### 6.1.10 Performance

La performance d'un système de contrôle d'accès détermine la capacité du système à traiter efficacement les requêtes qui lui sont envoyées. Théoriquement, la performance peut être déterminée à travers le calcul de la complexité algorithmique du modèle de contrôle d'accès. La performance est une propriété très critique dans les systèmes comportant plusieurs utilisateurs comme ceux de la santé. En cas d'urgence, le système doit répondre rapidement aux requêtes des médecins pour ne pas entraver le traitement des patients. L'implémentation de SGAC utilisée est celle mise en oeuvre par les développeurs de ce modèle. Pour le modèle XACML, c'est l'implémentation *Balana* qui a été utilisée. Les tests sont faits sur une même machine dans laquelle tourne Windows server 2008 R2. Cette machine a une mémoire vive de 4.00 GB et un processeur de 2.67 GHz. Pour effectuer les tests, nous avons fait un programme qui génère des règles et des requêtes de façon aléatoire. Le programme développé crée d'abord les requêtes et les règles dans le langage de SGAC et traduit ensuite les requêtes et les règles SGAC dans le langage XACML.

## 6.1. CRITÈRES D'ÉVALUATION

### Description de l'algorithme de génération des règles et des requêtes

L'objectif de ce programme est de créer un nombre très élevé de règles et de requêtes qui seront utilisées pour tester la performance des systèmes SGAC et XACML. Pour bien comprendre cette description du programme de génération des règles et des requêtes, le lecteur doit avoir lu au préalable les descriptions des modèles XACML et SGAC présentées respectivement dans les chapitres 4 et 5. Le programme de génération des règles et des requêtes prend en entrée cinq entiers naturels qui sont : la *profondeur des graphes des ressources et des sujets*, le *nombre de fils pour chaque nœud*, le *nombre de requêtes*, le *nombre de règles applicables par requête* et le *nombre de règles non applicables par requête*. Le programme offre en sortie six fichiers (*resources.txt*, *subjects.txt*, *SGAC\_rules.txt*, *SGAC\_request.txt*, *XACML\_Policy.txt*, *XACML\_Request.txt*). Le fichier *resources.txt* contient le graphe des ressources ; le fichier *subjects.txt* contient le graphe des sujets ; le fichier *SGAC\_rules.txt* contient les règles SGAC, le fichier *SGAC\_request.txt* contient les requêtes SGAC, le fichier *XACML\_Policy.txt* contient les politiques XACML et le fichier *XACML\_Request.txt* contient les requêtes XACML. Pour chaque graphe, le nombre de nœuds générés est calculé suivant la formule :  $(f^p - 1)/(f - 1)$ , où  $f$  = nombre de fils, et  $p$  = profondeur.

Le programme commence par la génération du graphe des ressources et du graphe des sujets. Après la génération des graphes, le programme rend certains nœuds du graphe des ressources paramétrables. Les nœuds paramétrables sont des nœuds qui peuvent être instanciés avec des valeurs. Par exemple, le nœud *patient* du graphe des ressources peut prendre les valeurs : *Simon*, *Romain*. Après avoir généré les graphes, le programme génère ensuite les requêtes et enfin les règles à partir des requêtes. Une requête porte sur une feuille du graphe des ressources et sur une feuille du graphe des sujets. Pour générer une requête  $Q$ , le programme procède de la façon suivante :

1. il choisit de façon aléatoire une feuille dans le graphe des ressources, par exemple, la feuille  $R\_12\_F$  ;
2. il détermine l'ensemble  $A$  des ancêtres de  $R\_12\_F$  ;
3. il affecte des valeurs aux nœuds paramétrables appartenant à  $A$ . Les valeurs affectées aux nœuds paramétrables permettent d'identifier  $R\_12\_F$  de façon unique.  $R\_12\_F$  correspond par exemple au test  $VIH\_123$  du patient *Simon* ;

## CHAPITRE 6. IDENTIFICATION DES CRITÈRES D'ÉVALUATION ET COMPARAISON DE RBAC, XACML ET SGAC

4. il choisit une feuille dans le graphe des sujets, par exemple, la feuille  $S_{15}$ . Cette feuille peut correspondre à l'infirmière *Alice* ;
5. il détermine l'ensemble  $B$  des nœuds ancêtres de  $S_{15}$ .

Pour générer une règle applicable à la requête  $Q$ , le programme procède comme suit :

1. il choisit comme ressource de la règle,  $R_{12}_F$  ou un nœud dans  $A$ . Les nœuds paramétrables portent les valeurs qui leur ont été assignées lors de la génération de la requête  $Q$  ;
2. Il choisit comme sujet,  $S_{15}$  ou un nœud dans  $B$  ;
3. il choisit de façon aléatoire une valeur entre *interdiction* et *permission*, comme modalité de la règle applicable ;
4. il choisit de façon aléatoire un entier naturel comme priorité de la règle applicable ;
5. il boucle sur les instructions précédentes autant de fois qu'il faut de règles applicables à  $Q$ . Pour générer les règles applicables à la prochaine requête  $Q2$ , le programme choisit les valeurs pour des nœuds paramétrables dans une tranche différente de celle de  $Q$ , de telle sorte que les règles applicables à  $Q$  ne soient pas applicables à  $Q2$ . Exemple  $R_0 = 3$  pour  $Q$  et  $R_0 = 14$  pour  $Q2$ .

Pour générer les règles non applicables à  $Q$ . Le programme procède comme suit :

1. il détermine l'ensemble  $X$  qui est le complémentaire de  $A$  dans l'ensemble  $Gr$  des nœuds ressources ;
2. il détermine l'ensemble  $Z$  qui est le complémentaire de  $B$  dans l'ensemble  $Gs$  des nœuds sujets ;
3. il choisit de façon aléatoire un élément dans  $X$  comme ressource de la règle non applicable. Si l'élément choisi est paramétrable il lui affecte une valeur ;
4. il choisit de façon aléatoire un élément de l'ensemble  $Z$  comme sujet de la règle non applicable ;
5. il choisit de façon aléatoire une valeur entre *interdiction* et *permission* comme modalité de la règle non applicable ;
6. il choisit de façon aléatoire un entier naturel comme priorité de la règle non applicable ;



## 6.1. CRITÈRES D'ÉVALUATION

7. Il boucle sur les instructions précédentes autant de fois qu'il faut de règles non applicables à  $Q$ .

Après avoir généré les requêtes et les règles SGAC, le programme les traduit en requêtes et règles XACML.

La fonction qui traduit une requête SGAC en une requête XACML, prend en entrée le format d'une requête XACML et la requête SGAC à traduire. Elle écrit chaque valeur d'attribut de la requête SGAC (*sujet*, *ressource*, *action*) à l'endroit correspondant à sa catégorie dans le format de la requête XACML.

La fonction qui traduit les règles SGAC en XACML, prend en entrée l'ensemble des règles SGAC et un format de document de politique XACML. Elle offre en sortie un document XACML contenant la traduction des règles en XACML. Les modèles XACML et SGAC n'ont pas le même algorithme d'évaluation des requêtes. En cas d'apparition d'un conflit entre deux règles lors de l'évaluation d'une requête, SGAC choisit la règle devant servir à rendre la décision suivant cet ordre : la *priorité* d'abord, ensuite le *sujet le plus spécifique* et enfin l'*interdiction*. XACML utilise des algorithmes de combinaison pour sélectionner la règle devant servir à rendre la décision. Aucun des algorithmes de combinaison de XACML n'est spécifié pour rendre une décision de la même manière que fait SGAC. Afin d'obtenir de la part des deux systèmes XACML et SGAC les mêmes résultats d'évaluation des requêtes, il convient d'ordonner les règles et politiques XACML suivant la *priorité*, la *spécificité des sujets* et l'*interdiction*, et choisir un algorithme de combinaison qui respecte cet ordre. L'algorithme de cette fonction se présente comme suit :

1. tri les règles SGAC par ordre de priorité ;
2. tri les règles de même priorité, suivant le niveau de profondeur de chaque sujet dans le graphe des sujets. Les règles ayant les sujets les plus profonds sont classées avant celles qui ont les sujets les moins profonds ;
3. tri les règles qui ont des sujets situés à une même profondeur, en classant d'abord les interdictions avant les permissions ;
4. pour la conversion proprement dite de la règle SGAC en XACML, la fonction utilise le format de document XACML et affecte à l'intérieur les valeurs aux attributs suivants :

## CHAPITRE 6. IDENTIFICATION DES CRITÈRES D'ÉVALUATION ET COMPARAISON DE RBAC, XACML ET SGAC

- à l'identifiant de la règle XACML (*ruleID*), est affecté le numéro de la règle SGAC,
- à la ressource de la target de la règle XACML, est affectée la ressource de la règle SGAC. Avant d'affecter la valeur de la ressource, la fonction de traduction appelle une autre fonction qui transforme la ressource sous la forme d'un chemin *path*. Cette transformation est nécessaire pour prendre en compte la hiérarchie des ressources dans XACML. Les champs qui entrent dans la composition du chemin *path* sont constitués des ancêtres de la ressource de la règle SGAC,
- au rôle du sujet dans la condition de la règle XACML, est affecté le sujet de la règle SGAC.

Les règles et les requêtes générées servent à mesurer les performances des systèmes XACML et SGAC. Les tests sont faits sur la même machine. Les résultats obtenus se présentent comme suit :

- les deux systèmes choisissent la même règle pour rendre la décision lorsqu'ils évaluent la même requête ;
- le temps de traitement d'une requête par XACML est plus long que le temps de traitement de la même requête par SGAC. Quand le nombre de règles est élevé (exemple 100.000 règles) SGAC est en moyenne 100 fois plus rapide. La figure 6.1 compare les temps d'exécution des deux systèmes.
- le temps de traitement d'une requête par XACML augmente au fur et à mesure que le nombre de règles croît, alors que SGAC a un temps relativement constant.
- le temps de chargement des graphes par SGAC augmente au fur et à mesure que le nombre de nœuds croît. Lorsque le nombre de nœuds des graphes sujets et ressources est très élevé, SGAC met assez de temps pour charger les graphes. Par exemple, lorsque les deux graphes ont chacun 349525 nœuds (profondeur=10 et fils=4), SGAC met 28mns pour les charger. Ce problème sera corrigé dans les versions futures de SGAC, selon les auteurs.
- lorsque le nombre de règles est très élevé (exemple de 120.000 règles), XACML n'arrive pas à charger les règles. Nous utilisons l'implémentation Balana qui utilise elle aussi le PDP de *SUN*. Nous obtenons une erreur relative à une insuffisance de la taille de la mémoire de la JVM (*Java heap space*). Malgré

## 6.1. CRITÈRES D'ÉVALUATION

les augmentations faites de cette taille de mémoire, l'erreur n'est pas résolue. Cette erreur est peut-être résolue dans d'autres implémentations de XACML ; mais le chargement du fichier de politiques demeure très critique pour le bon fonctionnement de XACML.

Le tableau 6.1.10 présente les résultats des tests de performance effectués sur des implémentations des modèles SGAC et XACML.

N	Nœuds	Règles	Req	XACML				SGAC			
				Total	TM/r	rMax	rMin	Total	TM/r	rMax	rMin
1	1093	10000	10	1189	109	344	21	214	2	5	2
2	1093	20000	10	1381	128	529	39	251	2	3	2
3	1093	30000	10	2075	194	1116	57	510	1	3	1
4	5461	40000	10	2644	231	1043	68	915	1	3	<1
5	5461	50000	10	2931	263	1057	149	945	1	7	1
6	5461	60000	10	3404	302	1169	94	963	2	3	2
7	5461	70000	10	4001	361	2063	114	1000	1	2	1
8	21845	80000	10	4701	427	1756	173	7000	1	3	1
9	21845	90000	10	5738	523	2144	143	>7000	1	2	1
10	21845	100000	10	6773	617	3969	156	8000	2	4	1
11	21845	120000	10	-	-	-	-	>8000	2	5	2
12	21845	700000	10	-	-	-	-	11000	2	3	1
13	21845	1000000	10	-	-	-	-	13000	2	3	1
14	349525	200000	10	-	-	-	-	28.10 <sup>6</sup>	2	4	1

Les temps sont en millisecondes(ms).

**Req :** indique le nombre de requêtes exécutées.

**Total :** indique le temps d'exécution de l'ensemble des requêtes (y compris le temps de chargement des règles pour XACML et le temps de chargement des règles et des graphes pour SGAC).

**TM/req :** indique le temps moyen d'exécution d'une requête (temps des chargements non compris). Le TM/req multiplié par le nombre de requêtes donne le temps total d'exécution de l'ensemble des requêtes, sans le temps des chargements.

**rMax :** indique le temps de la requête dont le traitement a le plus duré (le temps des chargements non compris dans les mesures).

**rMin :** indique le temps de la requête dont le traitement a le moins duré (le temps des chargements non compris dans les mesures).

tableau 6.1 – Résultats des tests de performances des systèmes XACML et SGAC

## CHAPITRE 6. IDENTIFICATION DES CRITÈRES D'ÉVALUATION ET COMPARAISON DE RBAC, XACML ET SGAC

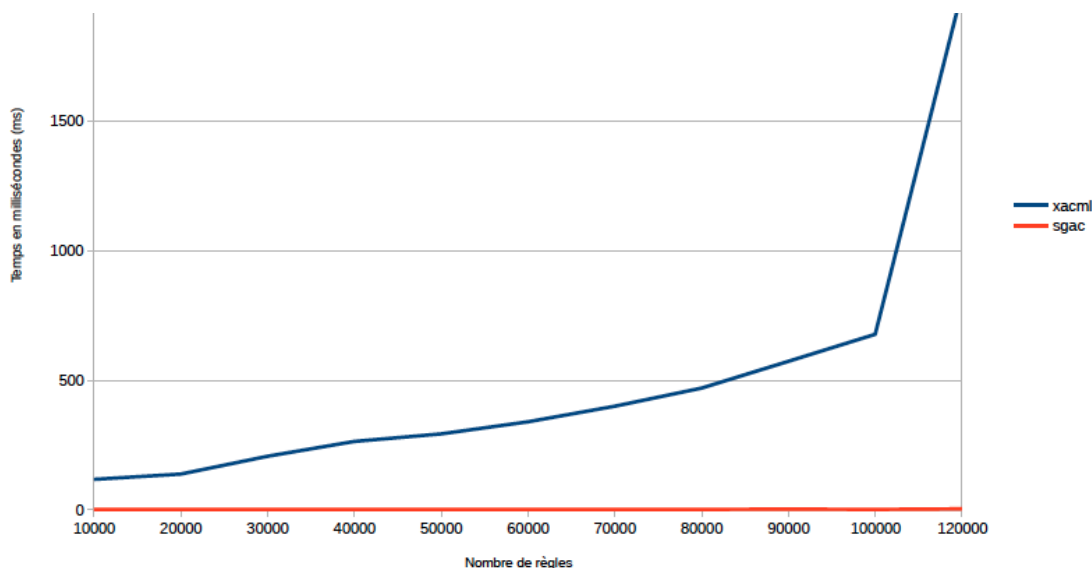


figure 6.1 – Comparaison entre XACML et SGAC, suivant le temps moyen d'exécution d'une requête après chargement des règles et des graphes

### 6.1.11 Expression des obligations

Les obligations ont une grande utilité dans la gestion du contrôle d'accès. Elles permettent de définir des actions qui seront exécutées lors de l'application de la réponse issue de l'évaluation d'une requête. La plupart des lois sur la protection de la vie privée des patients, recommandent que le patient soit informé lorsqu'un utilisateur accède à son dossier. Les obligations permettent de modéliser ces genres d'actions. XACML prend en compte les obligations. les obligations ne sont pas prises en compte dans RBAC. SGAC ne modélise pas les obligations.

## 6.2 Résumé de la comparaison de RBAC, XACML et SGAC

Le tableau 6.2 résume la comparaison entre RBAC XACML et SGAC.

De tout ce qui précède, RBAC n'est pas indiqué pour implementer le système du CHUS pour deux raisons principales. La première raison est que RBAC ne peut pas capturer toutes les règles de contrôle d'accès dans ce système. Ni les priorités entre les

## 6.2. RÉSUMÉ DE LA COMPARAISON DE RBAC, XACML ET SGAC

Critères	RBAC	XACML	SGAC
Expression des politiques	très peu expressif. Ne permet pas <ul style="list-style-type: none"> <li>— de modéliser les interdictions</li> <li>— les buts</li> <li>— les obligations</li> </ul>	très expressif, mais modélise difficilement la priorité entre règles	assez expressif, mais ne fait pas de distinction entre les opérations, ne modélise pas les obligations
Administration des politiques	pas facile dans les systèmes larges	difficile	facile
Le <i>Break the glass</i>	le modèle de base ne le permet pas	le modèle de base ne le permet pas	permet
Délégation des privilèges	les extensions du modèle permettent	ne permet pas	ne permet pas
Performance	bonne performance	la performance se dégrade avec l'augmentation des règles	bonne performance
Principe de moindre privilège	respecte	ne respecte pas	possible de faire respecter
Expression des conditions	ne permet pas	permet	permet
Contrainte de séparation des tâches	supporte	ne supporte pas	ne supporte pas
Prise en compte à la fois des interdictions et des permissions	modélise uniquement les permissions	modélise les deux	modélise les deux
Mécanisme de résolution de conflits	ne dispose pas	dispose	dispose
Expression des obligations	ne permet pas	permet	ne permet pas

tableau 6.2 – Comparaison entre RBAC, XACML et SGAC

règles, ni les règles dépendant d'un évènement dynamique ne peuvent être modélisées par RBAC. Des extensions de RBAC ont été proposées pour permettre à ce modèle de modéliser les évènements dynamiques. Parmi les extensions de RBAC, il n'y en a pas à notre connaissance, qui permette de modéliser les priorités entre les règles. La modélisation de la priorité entre les règles est nécessaire dans le système du CHUS où les règles émanent de plusieurs sources et peuvent être en conflit. La deuxième raison principale est la lourdeur que pourrait engendrer le système RBAC sur le plan administration des politiques. RBAC ne modélise pas la hiérarchie des ressources, ce

qui entraîne la multiplication du nombre de permissions à créer pour modéliser les règles. De plus, il n'est pas possible en RBAC d'utiliser les identités des utilisateurs dans les permissions. Même si la permission porte sur un seul utilisateur, il faut créer un rôle pour ce dernier, ce qui entraîne l'explosion de rôles.

XACML, malgré la richesse de ce langage, est le dernier choix auquel on pourrait faire recours parmi les trois modèles étudiés. Deux raisons principales expliquent pourquoi XACML n'est pas indiqué pour le système du CHUS. La première raison est liée à la performance du système XACML. La performance du système XACML se dégrade de façon linéaire avec l'augmentation du nombre de règles. Le nombre de règles dans le système du CHUS est estimé à un minimum de 700000 règles alors que, comme le montrent les tests effectués, PDP n'arrive pas à charger 120000 règles. Alex X. Liu et al. [39] ont proposé une solution pour améliorer les performances de XACML. Pour Alex X. Liu et al., la lenteur de XACML est due au fait que XACML utilise des fonctions qui comparent des chaînes de caractères. Leur solution consiste d'abord à convertir les chaînes de caractères en des valeurs numériques pour faciliter leur évaluation. Ils fournissent ensuite un algorithme qui normalise la structure de la politique XACML en convertissant tous les algorithmes de combinaison à l'algorithme *first-applicable* qui est rapide en exécution. Pour appliquer la solution de Alex X. Liu et al. en respectant les priorités entre les règles, les 700000 règles du CHUS doivent d'abord être ordonnées par priorité avant de les mettre à la disposition du système ; ce qui est très difficile à faire manuellement. La deuxième raison, est l'administration du système XACML qui est difficile. Le langage XACML est très complexe et difficile à maîtriser. Il est aussi difficile d'investiguer dans une politique XACML pour connaître les privilèges d'un utilisateur.

SGAC satisfait à beaucoup d'exigences du système du CHUS. Considérant le critère *administration des politiques*, SGAC offre plus de facilités que RBAC et XACML. Considérant le critère *expression des politiques*, SGAC est plus expressif que RBAC mais moins que XACML. SGAC a aussi une bonne performance. Cependant, SGAC ne modélise pas les opérations et les obligations. La réponse *interdiction* donnée par SGAC après évaluation d'une requête n'est pas explicite aux yeux de l'utilisateur. Le temps de chargement des graphes croît de façon quadratique avec l'augmentation du nombre de nœuds, ce qui peut dégrader la performance du système.

# Conclusion

Ces travaux ont abordé des questions importantes concernant les modèles de contrôle d'accès RBAC, XACML et SGAC. Chacun de ces trois modèles de contrôle d'accès a été étudié en profondeur ; leur application à un cas d'étude du contrôle d'accès tiré du réseau de santé de la région de Sherbrooke, a permis de dégager les limites de chacun d'eux.

RBAC ne permet pas de modéliser toutes les règles du contrôle d'accès du cas étudié. Les règles dépendant d'un événement dynamique ne sont pas modélisables en RBAC. Il y en a de même pour les interdictions qui ne sont pas modélisables en RBAC. L'impossibilité de modéliser ces règles, peut conduire à la violation de la politique de contrôle d'accès. Aussi, la hiérarchie des ressources n'est pas considérée dans la modélisation des règles en RBAC, ce qui entraîne la multiplication des permissions, et le risque que les règles préalablement définies ne s'appliquent pas aux nouvelles données. XACML est un langage complexe qui nécessite des connaissances techniques approfondies pour le maîtriser. L'administration des politiques est difficile et la délégation des permissions est impossible en XACML. Les performances d'un système XACML se dégradent de façon linéaire avec l'augmentation du nombre de règles. SGAC ne modélise pas les opérations et les obligations.

Les modèles ont été évalués sur la base des critères identifiés en regard des exigences du contrôle d'accès dans le domaine de la protection des dossiers médicaux électroniques. Il ressort de la comparaison qui s'en est suivi, que SGAC satisfait plus aux exigences identifiées dans ce domaine. Le modèle SGAC pourrait alors constituer une base d'étude pour le développement d'un modèle de contrôle d'accès plus générique.

## CONCLUSION



# Annexe A

## Code du PEP et du Context Handler

```
public class Health {

    private static Balana balana;
    public static void main(String[] args){
        String userName = null;
        initBalana();

        // UserRequest est le nom du fichier
        //contenant la requête de l'utilisateur
        //createXACMLRequest est une fonction du
        //Context Handler qui convertit la requête au format XACML
        String request = createXACMLRequest("UserRequest");
        PDP pdp = getPDPNewInstance();
        AttributeFinderHealth at = new AttributeFinderHealth();

        //la requête de l'utilisateur est reformulée en
        //transformant la valeur de la ressource
        request= at.findResource(request);

        //envoi de la requête au PDP pour évaluation
        String response = pdp.evaluate(request);
        System.out.println(response);
    }
}
```

## ANNEXE A. CODE DU PEP ET DU CONTEXT HANDLER

```
}

//Fonction d'initialisation de Balana qui indique
//l'endroit où sont enregistrées les politiques
private static void initBalana(){
    try{
        String policyLocation = (new File(".")).getCanonicalPath() +
            File.separator + "policies";
        System.setProperty(FileBasedPolicyFinderModule.
            POLICY_DIR_PROPERTY, policyLocation);
    } catch (IOException e) {
        System.err.println("Can not locate policy repository");
    }
    balana = Balana.getInstance();
}

//fonction qui crée une nouvelle instance du PDP,
//on indique au PDP les classes du PIP
private static PDP getPDPNewInstance(){

    PDPCConfig pdpConfig = balana.getPdpConfig();
    AttributeFinder attributeFinder = pdpConfig.getAttributeFinder();
    List<AttributeFinderModule> finderModules=attributeFinder.getModules();
    finderModules.add(new AttributeFinderHealth());
    attributeFinder.setModules(finderModules);

    return new PDP(new PDPCConfig(attributeFinder,
        pdpConfig.getPolicyFinder(), null, true));
}

/*cette fonction transforme la requête de l'utilisateur.
C'est une fonction du Context Handler.
La requête est déjà au format XACML.
Nous changeons juste les caractères de fin de ligne*/
```

```

public static String createXACMLRequest(String nomfichier){
    InputStreamReader flog = null;
    LineNumberReader llog = null;
    String myLine      = null;
    String myConcatLines = "";
    try{
        flog = new InputStreamReader(new FileInputStream(File_request) );
        llog = new LineNumberReader(flog);
        while ((myLine = llog.readLine()) != null) {
            myConcatLines += myLine + "\n";
        }
        }catch (Exception e){
            System.err.println("Error : "+e.getMessage());
            return null;
        }
    return myConcatLines;
    }
}

```

## ANNEXE A. CODE DU PEP ET DU CONTEXT HANDLER

# Annexe B

## Code du PIP

```
public class AttributeFinderHealth extends AttributeFinderModule {

    private URI defaultSubjectId;
    List<AttributeValue> attr = new ArrayList<AttributeValue>();
    public static String patient;

    private String url = "jdbc:postgresql://localhost:5432/Health";
    private String user = "user1";
    private String passwd = "pwd";

    public AttributeFinderHealth() {

        try {
            defaultSubjectId = new URI("urn:oasis:names:tc:xacml:1.0:
            subject:subject-id");
        } catch (URISyntaxException e) {

        }

    }

    @Override
    public Set<String> getSupportedCategories() {
        Set<String> categories = new HashSet<String>();
        categories.add("urn:oasis:names:tc:xacml:1.0:
```

## ANNEXE B. CODE DU PIP

```
        subject-category:access-subject");
        return categories;
    }

    @Override
    public Set getSupportedIds() {
        Set<String> ids = new HashSet<String>();
        ids.add("http://wso2.org/attribute/roleNames");
        return ids;
    }

    @Override
    public EvaluationResult findAttribute(URI attributeType, URI attributeId,
        String issuer, URI category, EvaluationCtx context) {

        EvaluationResult result = context.getAttribute(attributeType,
            defaultSubjectId, issuer, category);

        if(result != null && result.getAttributeValue() != null
            && result.getAttributeValue().isBag()){
            BagAttribute bagAttribute=(BagAttribute) result.getAttributeValue();
            if(bagAttribute.size() > 0){
                String userName = ((AttributeValue)
                    bagAttribute.iterator().next()).encode();
                findRole(userName);
            }
        }
        return new EvaluationResult(new BagAttribute(attributeType, attr));
    }

    @Override
    public boolean isDesignatorSupported() {
        return true;
    }

    //cette fonction se connecte à la base de données et
    //recherches les rôles de l'utilisateur
    private void findRole(String userName){
```

```

    try {
        Class.forName("org.postgresql.Driver");
        Connection conn=DriverManager.getConnection(url, user, passwd);
        Statement state = conn.createStatement();
        ResultSet result = state.executeQuery("SELECT distinct "+
            "professionnels.code_profession,
            professionnels.code_service, "+
            "professionnels.code_specialite,code_service2"+
            "FROM professionnels, service WHERE "+ "
            "service.code_service = professionnels.code_service AND "+
            "code_professionnel =' "+userName+"'");
        ResultSetMetaData resultMeta = result.getMetaData();

        while(result.next()){
            for(int i = 1; i <= resultMeta.getColumnCount(); i++)
                if (result.getObject(i) !=null)
                    attr.add(new StringAttribute(result.getObject(i).toString()));
            }
            result.close();
            state.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

//Cette fonction transforme la valeur de la ressource de la requête  
//suivant le format de la hiérarchie des ressources

```

public String findResource(String request){
    String resultat=null;
    String regex = "</AttributeValue>";

    Pattern p = Pattern.compile(regex);
    String[] items = p.split(request);
    String regex2 = "string\\>";

    Pattern p2 = Pattern.compile(regex2);
}

```

## ANNEXE B. CODE DU PIP

```
String[] items2 = p2.split(items[1]);
String valeur= items2[1];

try {
    ResultSet result = state.executeQuery("SELECT "+
        code_patient,code_service,code_groupe_donnees,"+
        code_sgrp_donnees,code_donnee,"+
        "id_donnee_patient FROM donnees_patients "+
        "WHERE id_donnee_patient ='"+valeur+"'");
    ResultSetMetaData resultMeta = result.getMetaData();

    while(result.next()){
        resultat = result.getObject(1).toString();
        patient=resultat;
        for(int i = 2; i <= resultMeta.getColumnCount(); i++)
            resultat = resultat + "/" + result.getObject(i).toString();
        }
        result.close();
        state.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    if(resultat == null)
        return request;
    else
        return request.replaceAll(valeur, resultat);
}
```



# Bibliographie

- [1] « Les lignes directrices de l'OCDE ».  
<http://www.oecdbookshop.org/oecd/index.asp?lang=fr>, dernier accès : 10 septembre 2015.
- [2] « RFC3986 ».  
<http://tools.ietf.org/html/rfc3986>, dernier accès : 10 septembre 2015.
- [3] « UMU XACML editor ».  
<http://umu-xacmleditor.sourceforge.net>, dernier accès : 10 septembre 2015.
- [4] Al-Kahtani M. A. et Sandhu R..  
« Rule-based RBAC with negative authorization ».  
*In 20th Annual Computer Security Applications Conference*, pages 405–415, IEEE, 2004.
- [5] Ferreira A., Cruz-Correia R., Antunes L. H. M., Farinha P., OLIVEIRA-PALHARES, E., Chadwick D. W. et Costa-Pereira A.  
« How to break access control in a controlled manner ».  
*19th IEEE International Symposium on Computer-Based Medical Systems, 2006*, pages 847–854, IEEE, 2006.
- [6] Jose Luis Fernandez ALEMAN, Inmaculada Carrion SENOR, Pedro Angel Oliver LOZOYA et Ambrosio TOVAL.  
« Security and privacy in electronic health records : a systematic literature review ».  
*Journal of Biomedical Informatics 46*, pages 541–562, 2013.

- [7] Ajit APPARI et Eric JOHNSON.  
« Information Security and Privacy in Healthcare : Current State of Research ». *International Journal of Internet and Enterprise Management*, pages 279–314, 2010.
- [8] Canadian Standards ASSOCIATION.  
« Les Lois fédérales du Canada sur la protection de la vie privée ». <http://www.parl.gc.ca/Content/LOP/ResearchPublications/prb0744-f.htm>, dernier accès : 10 septembre 2015.
- [9] AXIOMATIC.  
<http://www.axiomatics.com>, dernier accès : 10 septembre 2015.
- [10] AXIOMATIC.  
« Attribute Based Access Control (ABAC) the Best Cure for Sustainable eHealth Services. ». Rapport Technique, AXIOMATIC, 2010.
- [11] Joshi James BD, Bertino ELISA, Latif USMAN et Ghafoor ARIF.  
« A generalized temporal role-based access control model ». *IEEE Transactions on Knowledge and Data Engineering*,, pages 4–23, IEEE, 2005.
- [12] David F.C. BREWER et Michael J. NASH.  
« The Chinese Wall security policy ». *IEEE Symposium on Security and Privacy*, pages 215–228, 1989.
- [13] Inforoute Santé CANADA.  
« Dossier de santé électronique (DSE) Exigences en matière de protection de la confidentialité et de sécurité ». <https://www.infoway-inforoute.ca/fr/component/edocman/ressources/documents-techniques/295-exigences-pvps>, dernier accès : 10 septembre 2015.
- [14] Omar CHOWDHURY, Haining CHEN, Jianwei NIU, Ninghui LI et Elisa BERTINO.  
« On XACML’s Adequacy to Specify and to Enforce HIPAA ». *Proceedings of the 3rd USENIX conference on Health Security and Privacy*, pages 11–21, August USENIX Association, 2012.

## BIBLIOGRAPHIE

- [15] Abdallah A. E. et Takabi H..  
« Integrating delegation with the formal core RBAC model ».  
*Fourth International Conference on Information Assurance and Security, ISIAS'08*, pages 33–36, IEEE, 2008.
- [16] A.A. Abd EL-AZIZ et A. KANNAN.  
« A comprehensive presentation to XACML ».  
*Third International Conference on Computational Intelligence and Information Technology, 2013 (CIIT 2013)*, pages 155 – 161, IEEE Xplore, 2013.
- [17] Bell D ELLIOTT et La Padula Leonard J.  
« Secure computer system : Unified exposition and multics interpretation ».  
Rapport Technique, DTIC Document, 1976.
- [18] Yuan ERIC et Tong JIN.  
« Attributed Based Access Control (ABAC) for web services ».  
*Proceedings of the IEEE International Conference on Web Services*, pages 561–569, IEEE Computer Society, 2005.
- [19] Achim D. Brucker et HELMUT PETRITSCH.  
« Extending Access Control Models with Break-glass ».  
*Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 197–206, ACM, 2009.
- [20] David FERRAIOLO et Richard KUHN.  
« Role-based access controls ».  
*15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.
- [21] Ana FERREIRA, David CHADWICK, Pedro FARINHA, Ricardo CORREIA, Gansen ZAO, Rui CHILRO et Luis ANTUNES.  
« How to securely break into RBAC : the BTG-RBAC model ».  
*Computer Security Applications Conference, 2009 ACSAC'09*, pages 23–31, IEEE, 2009.
- [22] ANSI. American National Standard for INFORMATION TECHNOLOGY.  
« role based access control ».  
*INCITS 359-2012*, May 2012.

- [23] G. SCOTT GRAHAM et PETER J. DENNING.  
« Protection-principles and practice ».  
*Proceedings of the May 16-18, 1972, spring joint computer conference*, pages 417–429, ACM, 1972.
- [24] Tyrone GRANDISON et John DAVIS.  
« The impact of industry constraints on model- driven data disclosure controls ».  
*Proceedings of the 1st International Workshop on Model-Based Trustworthy Health Information Systems (MOTHIS) 2007*, 2007.
- [25] Diala Abi HAIDAR, Nora CUPPENS-BOULAHIA, Frederic CUPPENS et Herve DEBAR.  
« An Extended RBAC Profile of XACML ».  
*ACM Workshop on Secure Web Services (SWS)*, pages 13–22, 2006.
- [26] Andy HAN.  
« XACML is Growing Up ».  
<https://nextlabs.wordpress.com/2013/06/10/xacml-is-growing-up/>, dernier accès : 10 septembre 2015.
- [27] HL7.  
« Health Level Seven International ».  
<http://www.hl7.org/implement/standards/index.cfm>, dernier accès : 10 septembre 2015.
- [28] Vincent C. HU, David F. FERRAILOLO et D. Rick KUHN.  
« Assessment of Access Control Systems ».  
Rapport Technique, NIST, 2006.
- [29] INFOWAY.  
« Aperçu du DSE, du DME et du DSP ».  
<https://www.infoway-inforoute.ca/fr/ce-que-nous-faisons/la-santenumerique-et-vous/aperçu-du-dse-du-dme-et-du-dsp>, dernier accès : 10 septembre 2015.

## BIBLIOGRAPHIE

- [30] Canada Health INFOWAY.  
« Electronic Health Record (EHR) Privacy and Security Requirements », june 2015.
- [31] Ponemon INSTITUTE.  
« 2015 Cost of Data Breach Study : Impact of Business Continuity Management ». Rapport Technique, Ponemon Institute, june 2015, <http://www-03.ibm.com/security/data-breach/>, dernier accès : 10 septembre 2015.
- [32] Biba Kenneth J.  
« Integrity considerations for secure computer systems ». Rapport Technique, DTIC Document, 1977.
- [33] Xin JIN, Ram KRISHNAN et Ravi SANDHU.  
« A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC. ». *Data and Applications Security and Privacy XXVI*, pages 41–55, Springer, 2012.
- [34] Q. JIONG et M. CHEN-HUA.  
« Detecting and resolving constraint conflicts in role-based access control ». *Electrical and Control Engineering (ICECE), 2011 International Conference on. IEEE*, pages 5845–5848, 2011.
- [35] Patrick KIERKEGAARD.  
« Electronic health record : Wiring Europe’s healthcare ». *Computer law and security review*, pages 503–515, 2011.
- [36] Richard KISSEL.  
*Glossary of Key Information Security Terms*. DIANE Publishing, 2013.
- [37] Richard KUHN, Edward J. COYNE et Timothy R. WEIL.  
« Adding attributes to role-based access control ». *Computer*, pages 79–81, 2010.
- [38] Butler W. LAMPSON.  
« Protection ». *5th Princeton Symposium on Information Sciences and Systems*, pages 437–443, March 1971.

- [39] Alex X. LIU, Fei CHEN, JeeHyun HWANG et Tao XIE.  
« XEngine : A Fast and Scalable XACML Policy Evaluation Engine ».  
*In ACM SIGMETRICS Performance Evaluation Review*, pages 265–276, ACM, 2008.
- [40] Hasani S. M. et Modiri N..  
« Criteria Specifications for the Comparison and Evaluation of Access Control Models ».  
*International Journal of Computer Network and Information Security (IJCNIS)*, pages 19–29, 2013.
- [41] J. McLEAN.  
« The specification and modeling of computer security ».  
*Computer*, pages 9–16, 1990.
- [42] G. H. MOTTA et S. S. FURUIE.  
« A contextual role-based access control authorization model for electronic patient record ».  
*IEEE Transactions on Information Technology in Biomedicine*, pages 202–207, 2003.
- [43] Alan C. O’CONNOR et Ross J. LOOMIS.  
« 2010 Economic analysis of role-based access control ».  
Rapport Technique, NIST, RTI Project Number 0211876, 2010.
- [44] Ni Q., Bertino E. et Lobo J..  
« An obligation model bridging access control policies and privacy policies ».  
*Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 133–142, 2008.
- [45] Annanda RATH et Jean-Noel COLIN.  
« Access and usage control requirements for patient controlled records type of healthcare information system ».  
*The 7th International Conference on Health Informatics*, pages 331–336, HEALTHINF, 2013.
- [46] Annanda RATH et Jean-Noel COLIN.  
« Patient privacy preservation : P-RBAC vs OrBAC in patient controlled records

## BIBLIOGRAPHIE

- type of centralized healthcare information system. case study of wallon healthcare network, belgium. ».
- The Fourth International Conference on eHealth, Telemedicine, and Social Medicine eTELEMED 2012*, pages 111–118, IARIA, 2012.
- [47] L. ROSTAD et O. EDSBURG.  
« A study of access control requirements for healthcare systems based on audit trails from access logs ».  
*Computer Security Applications Conference, ACSAC'06. 22nd Annual*, pages 175–186, IEEE, 2006.
- [48] Pierangela SAMARATI et Sabrina De Capitani di VIMERCATI.  
« Access Control : Policies, Models, and Mechanisms ».  
*Foundations of Security Analysis and Design : Tutorial Lectures*, pages 137–196, Springer , 2001.
- [49] Ravi S. SANDHU.  
« Lattice-Based Enforcement of Chinese Walls ».  
*Computers and Security*, pages 753–763, 1992.
- [50] OASIS STANDARD.  
« eXtensible Access Control Markup Language (XACML) Version 3.0. ».  
2013.  
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html>., dernier accès : 10 septembre 2015.
- [51] V. SUHENDRA.  
« A Survey on Access Control Deployment ».  
*Security Technology*, pages 11–20, Springer, 2011.
- [52] SUN.  
« Implementation de XACML ».  
<http://sunxacml.sourceforge.net>, dernier accès : 10 septembre 2015.
- [53] OASIS XACML TC.  
« Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of XACML ».  
Rapport Technique, OASIS, june 2009, <http://docs.oasis-open.org/>

- [xacml/xspa/v1.0/saml-xspa-1.0.pdf](#), dernier accès : 10 septembre 2015.
- [54] OASIS XACML TC.  
« XACML v3.0 Core and Hierarchical Role Based Access Control (RBAC) Profile Version 1.0 ».  
Rapport Technique, OASIS, October 2014, <http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/xacml-3.0-rbac-v1.0.pdf>, dernier accès : 10 septembre 2015.
- [55] OASIS XACML TC.  
« XACML v3.0 Hierarchical Resource Profile Version 1.0 ».  
Rapport Technique, OASIS, May 2014, <http://docs.oasis-open.org/xacml/3.0/hierarchical/v1.0/xacml-3.0-hierarchical-v1.0.pdf>, dernier accès : 10 septembre 2015.
- [56] John TOLBERT.  
« XACML for Export Control and Intellectual Property Protection ».  
Rapport Technique, Boeing Company, 2009, <http://www.w3.org/2009/policy-ws/papers/Tolbert.pdf>, dernier accès : 10 septembre 2015.
- [57] Qihua WANG, Ninghui LI et Hong CHEN.  
« On the Security of Delegation in Access Control Systems ».  
*Proceedings of the 13th European Symposium on Research in Computer Security*, pages 317–332, Springer Berlin Heidelberg, 2008.
- [58] WSO2.  
« Implémentation de XACML ».  
<http://xacmlinfo.org/2012/08/16/>, dernier accès : 10 septembre 2015.
- [59] Xinwen ZHANG, Sejong OH et Ravi SANDHU.  
« PBDM : a flexible delegation model in RBAC ».  
*Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 149–157, ACM, 2003.