RÉSOLUTION DE CONFLITS DANS LA GESTION DU CONSENTEMENT DES DOSSIERS MÉDICAUX INFORMATISÉS.

par

Nghi Huynh

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, 14 novembre 2011

Sommaire

Dans le cadre des échanges d'information entre différentes entités, le cas du dossier médical partagé soulève de nombreux problèmes. En plus de la sécurité des informations, il faut ajouter le consentement du patient dans le processus d'échange de données. Le patient peut spécifier des règles qui vont à l'encontre des règles déjà existantes, pour diverses raisons. La résolution automatique des conflits que créent les règles du patient est délicate, et certaines solutions logicielles ont mis en place des mécanismes de résolution qui ne sont pas, à ce jour, suffisamment adéquats. Ce mémoire présente un algorithme de gestion de conflit prenant en compte une hiérarchie de profils et une hiérarchie de données. À partir d'une relation d'ordre définie sur des règles d'accès, l'algorithme reconstitue l'ensemble des données accessible au requérant suivant son profil. L'algorithme peut également être utilisé afin de constituer à partir d'une donnée, l'ensemble des profils autorisés à accéder à la donnée. Cette approche permet de ce fait de prendre en charge la notion de filtrage de données.

Mots-clés: dossier médical partageable ; dossier médical électronique ; gestion informatisée du consentement ; consentement ; dossier médical ; gestion de conflit de règles ; contrôle d'accès

Sommaire

Remerciements

Je souhaiterais exprimer toute ma gratitude à mon directeur de recherche, le Professeur Marc Frappier qui m'a offert l'opportunité d'effectuer une maîtrise au sein de son groupe de recherche et pour son implication dans ce travail de maîtrise.

Je remercie Émilie Au, ma soeur Anne Huynh et toute ma famille, pour leur soutien affectif et inconditionnel.

Je remercie les autres étudiants en maîtrise ou en doctorat qui m'ont accordé de leur temps et qui m'ont parfaitement intégré au sein de leur laboratoire.

Je suis très reconnaissant au laboratoire GRIL (Groupe de Recherche en Ingénierie du Logiciel) pour m'avoir accueilli et de m'avoir placé dans les meilleures conditions de travail.

Mes remerciements vont également à Thierry Le Ba qui a participé avec moi au projet que constituait ma maîtrise. Il en va de même au personnel du CHUS (Centre Hospitalier Universitaire de Sherbrooke), et plus particulièrement Lucie Frenette et Hassan Diab qui nous ont offert leur collaboration et qui ont toujours été disponibles pour répondre à nos questions.

Et enfin, mes remerciements vont aussi à l'ENSIIE, qui m'a permis d'effectuer ma troisième année ici à Sherbrooke.

Remerciements

Abréviations

BDM Banque de Données Médicales

CHUS Centre Hospitalier Universitaire de Sherbrooke

CLSC Centre Local de Service Communautaire

CRE Centre de Rééducation de l'Estrie

CRISP Cheapeake Regional Information System for Our Patient

DHIN Delaware Health Information Exchange

DMP Dossier Médical Partagé

DSP Directeur des Services Professionnels

DSQ Dossier Santé Québec

GMF Groupe de Médecine Familiale

HRB Health Record Bank

IETF Internet Engineering Task Force

IHIE Indiana Health Information Exchange

IUGS Institut Universitaire de Gériatrie de Sherbrooke

LIPIX Long Island Patient Information Exchange

MAeHC Massachussetts e-Health Collaborative

NHS National Health Service

NHS CRS National Health Service Care Records Service

PACS Picture Archiving and Communications System

PAP Policy Administration Point

PDP Policy Decision Point

PEP Policy Enforcement Point

PMO Programme Médicament de l'Ontario

PMS Policy Management Service

PIP Policy Information Point

RBAC Role Based Access Control

RIQI Rhode Island Quality Institute

SILO Système d'Information de Laboratoire de l'Ontario

VPPP Visualiseur des Profils Pharmaceutiques des Patients

Table des matières

Liste des figures

Liste des tableaux

Liste des programmes

Introduction

Contexte

Il est de plus en plus commun de voir ses données stockées de manière centralisée ou décentralisée sur des serveurs distants. Cette pratique se répand également dans le domaine médical, où la numérisation des données permet l'informatisation des dossiers médicaux. Les avantages à cette pratique sont nombreux :

- suivi plus complet et plus rapide au niveau du patient, car les informations sont plus facilement accessibles ;
- centralisation dans une base de données des résultats d'examens (examens Ultrasons, rayons X, IRM etc...) ;
- augmentation de la qualité des diagnostics et des traitements ;
- examens redondants évités pour le patient ;
- coût de consultation réduit et optimisation du ratio coût/bénéfice pour l'institution médicale ;
- communication aisée entre les différents établissements et/ou services.

Les Dossiers Médicaux Personnels (DMP) en plus du contrôle d'accès «classique», nécessitent un contrôle accru au niveau des données médicales à cause du cadre légal et des préjudices graves causés aux patients lors d'un bris de confidentialité. C'est pourquoi la mise en place d'un environnement et des mécanismes de gestion du consentement du patient lors de l'échange et/ou de l'utilisation des données devient alors cruciale.

Objectifs

L'objectif de nos travaux est de fournir un prototype fonctionnel au Centre Hospitalier Universitaire de Sherbrooke (CHUS) afin de pouvoir utiliser les Dossier Médicaux Personnels (DMP) dans le cadre défini par la loi. À l'heure actuelle, le CHUS possède déjà une solution pour les DMP nommée ARIANE qui contrôle les accès aux DMP selon le profil de l'utilisateur (celui-ci s'identifiant dans le système grâce à un système physique de clef électronique) : à chaque profil est attribué des droits d'accès, d'écriture et de modification. Les profils regroupent donc les différentes fonctions des intervenants. Notre solution logicielle doit prendre en compte le *consentement du patient*, fonctionnalité manquante d'ARIANE et devra aussi pouvoir se greffer à la solution du CHUS afin de ne pas perturber les pratiques existantes. Notre prototype doit également être capable de gérer des cas de conflits qui peuvent survenir (suite aux choix du patient) et opter pour le choix que ferait le patient. Finalement, notre solution doit être en mesure de filtrer les informations du DMP selon les règles instaurées par le patient, ARIANE n'étant capable de filtrer qu'une partie des données.

Méthodologie

Dans un premier temps, il est nécessaire de faire une recherche bibliographique afin de recenser les travaux effectués jusqu'à présent dans le domaine du DMP. Cela a permis de voir quels concepts sont sous-jacents au DMP.

- Qu'est ce que le *consentement* ?
- Comment doit-il être pris en compte ?
- Comment interagit-il avec les règles «classiques» d'accès ?
- Quelles sont les solutions mises en place ?
- Comment est géré le consentement ?

La recherche bibliographique permet de trouver les réponses à ces interrogations. Pour mieux appréhender le problème, il est nécessaire de comprendre le fonctionnement d'ARIANE. La coopération avec le service informatique et des archivistes du CHUS permet dans un second temps, en plus de la compréhension du fonctionnement d'ARIANE, de mettre en avant des cas de gestion du consentement pour lesquels notre

prototype doit fonctionner. On évalue ensuite plusieurs solutions présentes dans la littérature afin de les comparer, et d'en extraire les principes réutilisables pour notre projet. La phase de modélisation du problème quant à elle, permet de définir les patrons de règles et les concepts importants pour notre prototype.

Résultats

Un algorithme de gestion de conflit a été obtenu. Il s'appuie sur une hiérarchie de profils ainsi qu'une hiérarchie de données. L'algorithme permet de créer l'ensemble des données accessibles à un *sujet* à partir d'une requête contenant les informations *action*, *sujet de l'action*, *cible de l'action* que l'on note $<action(sujet, cible)>$. Cela permet de prendre en compte la notion de filtrage que l'on ne trouve ni en Ponder ni en Cassandra et qui n'est pas complet dans ARIANE.

Grâce à l'algorithme, il est également possible à partir de la même requête *<action(sujet,cible)>* de construire l'ensemble des profils qui ont accès à une certaine donnée. Cela permet de faire de la vérification au niveau du contrôle d'accès et pouvoir savoir si un patient peut, avec l'ensemble des règles qu'il a fixé, recevoir des soins normalement.

Structure du mémoire

Le mémoire est organisé comme suit. Le chapitre 1 introduit les notions nécessaires à la compréhension du mémoire ainsi que l'état de l'art. Celui-ci fait une étude des différents modes de gestion du consentement pour plusieurs pays tels le Royaume-Uni, les États-Uni et le Canada, et également de solutions de gestion du consentement. Le chapitre 2 est une étude détaillée d'une des solutions proposées. L'évaluation de Ponder permet d'appréhender la façon dont peuvent être résolus des conflits entre différentes règles d'accès. Le chapitre 3 traite de la modélisation de la solution, de la modélisation des différentes règles, issues du consentement ou issues du fonctionnement des institutions médicales. Le chapitre 4 présente l'algorithme de gestion de conflit conçu à partir de l'étude faite sur Ponder dans le chapitre 3 et des patrons

de règles faits dans le chapitre 4. Enfin, le chapitre 5 présente un cas d'étude pour illustrer l'algorithme de gestion de conflit. Une conclusion clôt ce mémoire.

Chapitre 1

État de l'art

Dans ce chapitre, il est fait un état de l'art présentant le dossier médical personnel et la législation l'entourant. En plus du fonctionnement du dossier médical personnel, l'état de l'art décrit également les modes de consentements adoptés dans divers pays montrant de ce fait qu'aucune solution adoptée par ces pays ne permet d'avoir une granularité assez fine dans la gestion du consentement pour les besoins du patient. L'état de l'art inclut en outre la description du projet conjointement mené avec le CHUS, pour préciser les contraintes que devra satisfaire le prototype, ainsi qu'une présentation de solutions existantes pour la gestion du consentement tels que XACML ou bien Ponder.

1.1 Dossier médical personnel

Dans cette section, nous nous intéressons au fonctionnement d'un dossier médical personnel (DMP), tel que définit par les différents textes légaux, afin de mieux comprendre le cadre dans lequel doit évoluer notre prototype. Ceux-ci définissent les droits de l'usager tels que le droit de consultation, le droit de rectification ou bien le droit de confidentialité. Le DMP doit obligatoirement respecter ces droits et permettre à l'usager de les exercer. Le personnel qui s'occupe du DMP doit quant à lui observer des devoirs fixés par ces mêmes textes de loi : devoir d'ouverture d'un dossier ou encore devoir de préservation de la confidentialité. En cas de manquement, des sanctions ont été prévue par ces textes. Ceux-ci prévoient également des cas où il est

possible d'accéder au dossier de l'usager sans son aval dans des conditions extrêmement précises : ces cas doivent respecter certain critères tel le critère de nécessité et permettent de parer à des situations d'urgences dans lesquelles par exemple l'usager ne serait pas en mesure d'accorder l'accès.

1.1.1 Présentation

Le regroupement des données médicales au sein d'un unique dossier soulève de nouvelles problématiques dont les principales sont la sécurité des informations et les problèmes légaux vis-à-vis de la vie privée des patients. Le cadre légal influe grandement notre travail. Selon la législation québécoise et canadienne, les droits d'un usager sur son dossier sont les suivants.

Droit de consultation

Sous réserve de certaines exceptions prévues par la Loi, « tout usager de 14 ans et plus a droit d'accès à son dossier» (*Loi sur les services de santé et les services sociaux* [7] (LSSSS [1]), article 17 ; *Code civil du Québec*[8] [2], article 38 ; *Loi sur l'accès aux documents des organismes publics et sur la protection des renseignements personnels* [6](LADOP) [3]), article 83).

Droit de rectification

« Toute personne peut faire corriger, dans un dossier qui la concerne, des renseignements inexacts, incomplets ou équivoques, elle peut aussi faire supprimer un renseignement périmé ou non, justifié par l'objet du dossier, ou formuler par écrit des commentaires et les verser au dossier» (*C.c. du Québec*, article 40, alinéa 1 ; *LADOP*. Article 89).

Droit de confidentialité

« Toute personne a droit au respect de sa vie privée » (*Charte des droits et libertés de la personne*, article 5).

1. http://www2.publicationsduquebec.gouv.qc.ca/dynamicSearch/telecharge.php?type=2&file=/S_4_2/S4_2.htm
2. http://www2.publicationsduquebec.gouv.qc.ca/dynamicSearch/telecharge.php?type=2&file=/CCQ/CCQ.html
3. http://www2.publicationsduquebec.gouv.qc.ca/dynamicSearch/telecharge.php?type=2&file=/A_2_1/A2_1.html

1.1. Dossier médical personnel

> « Toute personne a droit au respect de sa réputation et de sa vie privée. Nulle atteinte ne peut être portée à la vie privée d'une personne sans que celle-ci ou ses héritiers y consentent ou sans que la loi l'autorise » (*C.c. du Québec*, article 35).
>
> « Le dossier d'un usager est confidentiel et nul ne peut y avoir accès, si ce n'est avec le consentement de l'usager ou de la personne pouvant donner un consentement en son nom, sur l'ordre d'un tribunal ou d'un coroner dans l'exercice de ses fonctions ou dans le cas où la présente loi prévoit que la communication de renseignements contenus dans le dossier peut être requise d'un établissement » (*LSSSS*, article 19, par.1).

Quant au personnel, il doit également suivre des obligations qui sont accompagnées de sanctions, comme tout devoir :

Devoir d'ouverture d'un dossier

> «Un établissement doit tenir un dossier sur chacun des bénéficiaires qui en obtient des services... les renseignements exigés du bénéficiaire en vertu de l'article 23 sont conservés au dossier» (*Règlement sur l'organisation et l'administration des établissements*[4], article 50).

Devoir concernant le contenu du dossier

> Un établissement « ne peut recueillir que les renseignements pertinents à l'objet... du dossier» (*C.c. du Québec*, article 37).

Devoir de préservation de la confidentialité

> Les renseignements contenus au dossier sont décrétés confidentiels de sorte que le personnel « ne peut sans le consentement de l'intéressé ou l'autorisation de la loi, les communiquer à des tiers » (*C.c. du Québec*, article 37 ; LSSSS, article 19 ; *LADOP*, article 53).

Sanctions en cas de manquement à la confidentialité

> Si un membre du personnel manque à la confidentialité, les sanctions suivantes sont alors possibles :

4. http://www2.publicationsduquebec.gouv.qc.ca/dynamicSearch/telecharge.php?type=2&file=%2F%2FS_5%2FS5R3_01.htm

- condamnation, au civil, à réparer le préjudice (*C.c. du Québec*, article 1457) ;
- amende ou suspension du droit d'exercice si l'employé est membre d'une corporation professionnelle (voir les lois qui régissent les différentes corporations) ;
- suspension sans solde ou congédiement (selon la gravité de la faute) imposé par l'employeur ;
- amende de 200 $ à 1000 $ (*LADOP*, article 159).

1.1.2 Exceptions spécifiées par la Loi

La loi prévoit des cas d'exceptions où l'usager peut voir ses informations médicales divulguées. Dans ces cas là, la demande d'accès au dossier doit respecter les points suivants.

Critère de nécessité : c'est le critère le plus fondamental, chaque demande d'information doit nécessairement satisfaire ce critère.

Communication des informations à des fins précises : les informations obtenues doivent être divulguées dans un but valable et précis.

Respect des normes de sécurités et des règlements du ministre : même dans un cas d'exception, la confidentialité de l'information doit être préservée.

À partir de ces principes, la Loi a répertorié quelques exceptions (visibles dans les articles de la LSSSS).

- Cas de communication du dossier de l'usager sans son consentement :
  - mise à jour des fichiers et des index locaux,
  - vérification d'admissibilité des usagers,
  - exercice d'un mandat ou exécution d'un contrat de service,
  - sondages par un organisme d'accréditation reconnu aux fins de l'émission d'un agrément d'un établissement,
  - déploiement des systèmes partagés d'archivage et de communication des examens d'imagerie,
  - ententes pour dispense de certains services de santé ou de services sociaux,
  - ententes pour distribution automatisée de médicaments.

- Cas d'utilisation de renseignements contenus au dossier de l'usager sans son consentement :
  - utilisation par un établissement des nom, prénom et adresse d'un usager pour inviter celui-ci à souscrire un don,
  - utilisation par un établissement des nom, prénom et numéro de téléphone d'un usager pour la réalisation d'un sondage.

Il y a obligation de divulgation par exemple [5] :

- au tribunal, interrogé par l'avocat de son patient, le médecin appelé à témoigner peut briser le seau de la confidentialité car le patient a alors renoncé implicitement au secret professionnel. Par contre sans l'aval de son patient, il ne peut pas discuter du cas du patient ni fournir des pièces du dossier médical ;
- dans le cadre de la Protection de la jeunesse, si un médecin a un motif raisonnable de penser que la sécurité d'un enfant est compromis, il est tenu d'en informer les autorités.

Il y a autorisation de divulgation par exemple [6] :

- dans le cas d'un *tier en danger*, un organisme peut divulguer des informations personnelles «à une personne à qui cette communication doit être faite en raison d'une situation d'urgence mettant en danger la vie, la santé ou la sécurité de la personne concernée» ;
- dans le cadre de *statistiques*, un organisme peut divulguer des informations personnelles «à une personne qui est autorisée par la Commission d'accès à l'information, conformément à l'article 125 de la LADOP, à utiliser ce renseignement à des fins d'étude, de recherche ou de statistique» ;
- afin de *prévenir un crime*, un organisme peut divulguer des informations personnelles «à un organisme qui, en vertu de la loi, est chargé de prévenir, détecter ou réprimer le crime ou les infractions aux lois, si le renseignement est nécessaire aux fins d'une poursuite pour infraction à une loi applicable au Québec».

5. http://www.avocat.qc.ca/public/iidossiermedical2.htm
6. http://www.avocat.qc.ca/public/iidossiermedical3.htm et http://www2.publicationsduquebec.gouv.qc.ca/dynamicSearch/telecharge.php?type=2&file=/A_2_1/A2_1.html

1.2 Consentement

Cette section a pour but de présenter les différentes façon de modéliser le consentement, et quelles solutions ont adoptés des pays comme les États-Unis, le Canada, le Royaume-Uni, le Pays-Bas ou encore la Suède.

Cette section s'appuie sur un rapport produit par Melissa M.Goldstein pour «Office of the National Coordinator for Health Information Technology» [10] et le rapport produit par Joy Pritts et Kathleen Connor pour le SAMHSA (Substance Abuse and Mental Health Services Administration) du département américain de la santé[17].

Parmi les différents modèles étudiés, on peut distinguer cinq principaux modèles de consentements utilisés.

Aucun consentement requis : l'accès aux données médicales ne nécessite pas le consentement du patient ; ce mode est utilisé dans certains états américains (Indiana).

Opt-in : le consentement du patient est requis pour accéder à ses données ; toutefois, le patient ne peut spécifier de restriction particulière sur qui peut accéder ou sur ce qui peut être accédé.

Opt-out : le consentement du patient n'est pas requis pour accéder à ses données ; le patient peut se retirer complètement et interdire tout accès à ses données ; le patient ne peut spécifier de restriction particulière sur qui peut accéder ou sur ce qui peut être accédé. C'est le modèle retenu pour le DSQ.

Opt-in avec restrictions : le consentement du patient est requis pour accéder à ses données ; le patient peut spécifier des restrictions particulières sur qui peut accéder ou sur ce qui peut être accédé.

Opt-out avec restrictions : le consentement du patient n'est pas requis pour accéder à ses données ; le patient peut spécifier des restrictions particulières sur qui peut accéder ou sur ce qui peut être accédé.

On définit un *consentement implicite* comme étant un consentement que le patient n'a pas expressément donné. Généralement, le *consentement implicite* vient du fait qu'il faut nécessairement le consentement de l'usager afin de mettre en œuvre la volonté du patient : par exemple, lors d'une visite médicale le patient, afin de recevoir des

soins, consent *de manière implicite* à autoriser le personnel soignant à accéder à son dossier dans le cadre des soins qu'il reçoit. C'est donc équivalent à de l'*opt-out*.

1.2.1 Aux États-Unis

Aux États-Unis, la législation se fait au niveau des états ce qui donne divers modèles de DMP : le consentement peut être requis dans un état pour la consultation du dernier épisode alors qu'il ne l'est pas dans un autre état.

Les états peuvent utiliser plusieurs modèles suivant les cas : par exemple, le consentement peut ne pas être requis pour la sauvegarde des informations alors qu'il l'est pour la consultation de ces informations.

Voici les différents états américains étudiés :

1. le Delaware avec Delaware Health Information Exchange (DHIN) ;
2. l'Indiana avec Indiana Health Information Exchange (IHIE) ;
3. le Maryland avec Chesapeake Regional Information System for Our Patients (CRISP) [7] ;
4. le Massachussetts avec Massachussetts e-Health Collaborative (MAeHC) ;
5. New York avec Long Island Patient Information Exchange (LIPIX) [8], HEALTHeLINK [9], Souther Tier Health Link [10] (il existe 9 organismes régionaux) ;
6. Rhode Island avec RIQI (Rhode Island Quality Institute [11]) ;
7. Washington avec des banques de données médicales («*Health Record Bank*») tel que *Microsoft Health Vault*.
8. Tennessee et Virginie : Carespark [12] ;

Pas de consentement

Dans l'Indiana, les données sont automatiquement échangées sans le consentement du patient. Ces données ne sont donc soumises qu'à l'éthique et à la déontologie des

7. http://www.crisphealth.org/
8. http://www.lipix.org/
9. http://www.wnyhealthelink.com/
10. http://www.sthlny.com/
11. http://www.riqi.org/matriarch/default.asp
12. http://www.carespark.com/

praticiens.

Il existe également une partie du dossier clinique au Delaware qui ne nécessite pas le consentement du patient : les données sont sans son consentement ajoutées à son dossier médical.

De la même manière qu'en Indiana, les lois des états tels que la Virginie ou bien le Tennessee n'interdisent pas les échanges d'informations cliniques sans consentement exprès du patient dans le cadre de traitements ou d'autres cadres spécifiquement prévus par la loi.

Opt-out

L'opt-out est un modèle largement employé du fait qu'il fournit aux intervenants un accès par défaut aux dossiers médicaux. Dans celui en fonction au Delaware, il est possible de se retirer complètement du programme d'échange des informations cliniques, ce que n'a fait aucun patient du Delaware (encore faut-il que le patient sache qu'il en a la possibilité).

Au Maryland, les patients qui participent au CRISP [13] (Chesapeake Regional Information System) participent en fait à deux échanges :

- programme d'échange de données dans les hôpitaux du comté de Baltimore (pour l'historique de soin) ;
- programme d'échange de données dans ceux du comté de Montgomery (pour certaines informations cliniques).

Par rapport au Delaware, il est plus aisé de pratiquer son droit de retrait (il suffit juste d'un simple coup de téléphone). De même si le patient veut réintégrer le programme, il lui suffit d'appeler ou bien de remplir un formulaire. CareSpark [14](programme d'échange au niveau régional des informations médicales à but non lucratif), qui opère dans des comtés du Tennessee et de Virginie a choisi l'opt-out éclairé : les informations ne sont pas collectées tant que le patient n'a pas eu suffisamment d'informations au sujet du programme d'échange. La gestion du consentement est laissée quant à elle aux différents fournisseurs de soins qui peuvent très bien vouloir avoir recours au consentement du patient exprès pour les échanges comme ils peuvent échanger les

13. http://www.crisphealth.org/
14. http://www.carespark.com/dev/

informations sans son consentement (la législation des états le permet).

Opt-in

Dans certains états, une gestion du consentement plus granulaire est requise par la législation. Ainsi à Rhode Islands, le Rhode Islands Quality Institution (RIQI) utilise l'opt-in avec restrictions, les choix étant :

- permettre l'accès aux données à toutes les organisations dispensatrices de soins ;
- autoriser certains fournisseurs de soins seulement ;
- autoriser le cas par défaut avec accès temporaire aux informations par des gens habilités dans des cas d'exceptions (urgences et cas non prévus).

Bien que selon les médecins l'opt-in soit plus restrictif que l'opt-out, dans le Massachusetts le taux de patients ayant choisi de faire un opt-in a dépassé les 90% en novembre 2008. Un patient peut choisir quelles entités auront accès à son dossier lors de ses visites mais ne peut choisir quelles parties du dossier seront visibles.

New York a également choisi l'opt-in : le fournisseur de soin obtient le consentement du patient à un point de santé, l'échange étant permis via un formulaire, qui comprend un consentement pour les différents fournisseurs de soins, accessible au point de santé ou en ligne.

Parmi les formes d'opt-in on peut noter qu'il apparait une forme différente des autres. Jusqu'alors, les modèles étaient centrés sur le dossier du patient : le patient choisissait via un intermédiaire qui pouvait avoir accès à son dossier. Il existe également une autre forme de gestion du consentement : celle de la gestion du consentement du patient par le patient. C'est le principe des banques de données médicales (BDM) («*Health Record Bank*») qui sont mises en place dans des États tels Washington : le patient crée un compte personnel en utilisant des outils en ligne tels que *Microsoft Health Vault*[15] et *Google Health*[16] et gère lui même les accès à ses données sur son compte. Washington a dans cette optique mis en place des projets pilotes[17] de BDM.

15. Politique de protection de la vie privée : https://account.healthvault.com/help/en-US/default.htm
16. Politique de protection de la vie privée : http://www.google.com/intl/en-US/health/privacy.html
17. http://www.hca.wa.gov/hit/documents/straw_concept_final_draft032108.pdf

Modalité du consentement

Généralement, le consentement est obtenu à un centre de soin avec du personnel assistant afin d'informer le patient, l'orienter. D'autres formes de consentement, en dehors des organismes de la santé sont possibles : le consentement est généralement dans ces cas-là recueilli par téléphone ou grâce à Internet. On peut distinguer :

- le consentement obtenu en une fois ;
- le consentement obtenu avec multiples interactions avec l'utilisateur.

Concernant la portée du consentement, on distingue :

- le consentement global (englobe tous les cas) ;
- le consentement détaillé (certains cas sont explicités, les types de données partagées sont définis etc...).

Parmi les différentes solutions étudiées, le système de banque de données (BDM) de Washington est le seul système qui permet au patient de masquer dans les échanges leurs données par type de données : la limitation provient des solutions logicielles utilisées par le patient. Par exemple, parmi les deux BDM les plus connus, à savoir *Microsoft Health Vault* et *Google Health*, le niveau de granularité n'est pas le même : il n'était pas possible de masquer les données par types de données via Google Health. En 2011, Google a choisi d'interrompre son programme de santé faute d'utilisateurs. [18]

1.2.2 Au Royaume-Uni

Au Royaume-Uni, les patients seront en mesure de masquer les informations sensibles contenues dans leur dossier médical. Ils pourront demander à ce que certaines informations de leur dossier médical ne soient seulement accessibles qu'avec leur consentement : cet opération est appelé «scellement» (*sealing*). Comme un patient pourrait sceller des informations dans une «enveloppe scellée» (*sealed envelope*) [14] [19], un praticien peut également sceller certains types d'informations à l'insu du patient. Le modèle de consentement est donc un opt-out avec restrictions :

18. http://www.zdnet.fr/actualites/google-health-et-powermeter-arret-annonce-faute-d-utilisateurs-39762001.htm
19. http://www.connectingforhealth.nhs.uk/systemsandservices/infogov/confidentiality/sealedpaper.pdf

- un patient peut consentir ou refuser [20] au partage de ses informations cliniques avec le NHS CRS (National Health Service Care Records Service).
- un patient peut choisir des informations à masquer.

De plus le NHS CRS propose des contrôles d'accès. Les accès aux dossiers médicaux électroniques se font en trois temps :

1. Authentification : utilisation de la carte à puce pour authentifier chaque employé de la NHS avec un login.
2. Relation directe : afin d'accéder au dossier d'un patient, il faut posséder un lien direct («*legitimate relationships*») comme lorsque le requérant fait partie de l'équipe médicale chargée du patient.
3. Contrôle d'accès basé sur les rôles (RBAC)[9] : la quantité d'information visible est restreinte par le rôle que possède le requérant, afin de satisfaire le principe de nécessité.

Le patient ne peut pas empêcher la sauvegarde des informations dans son dossier médical.

1.2.3 Au Pays-Bas

Au Pays-Bas, le modèle de consentement retenu est en fait un modèle basé sur l'opt-out. La granularité dépend des relais d'informations connectés à la base de données nationale. En effet, les données du patient sont conservées à l'échelle régionale (les échanges régionaux électroniques d'informations médicales sont déjà mis en place) et sont disponibles aux praticiens accrédités du pays entier. Bien que le consentement soit un *opt-in complet* lors de l'acceptation du traitement médical, le patient a la possibilité de masquer ses données en fonction des fournisseurs, du type de données, du type de soin, mais également de se retirer des échanges : c'est donc un *consentement implicite*. À l'heure actuelle, beaucoup de DMP sont utilisés mais ne correspondent pas aux standards mis en place par le gouvernement à travers le projet LSP, «*Landelijk SchakelPunt*» (ou littéralement «*Point de connexion national*», autrement appelé

20. formulaire d'opt-out disponible ici :http://www.connectingforhealth.nhs.uk/systemsandservices/scr/staff/aboutscr/comms/pip/optout.pdf

Infrastructure Informatique Nationale Néerlandaise d'Echange d'Informations Médicales). L'appel d'offre a été remportée en 2005 par InterSystems qui a mis au point la plateforme «Ensemble» sur laquelle repose LSP [21]. Le projet est de faire un méta-index : les informations ne seront pas centralisées mais indexées, c'est la raison pour laquelle les données sont enregistrées à l'échelle régionale.

Néanmoins, du fait des différents DMP sur lequel repose le projet national, le projet est critiqué pour la sécurité des informations.

1.2.4 En Suède

En Suède, le système médical facilite la mise en place des DMP : la plupart des informations sont déjà sous forme numérique. Ayant la volonté d'améliorer la qualité des soins reçus par les suédois, le parlement a fait voter une loi qui permet au patient de choisir qui peut accéder à son dossier et qui permet également de choisir les données qui seront transmises dans l'échange. Ce projet, qui a pour nom «*Nationell Patientöversikt*» (NPÖ) permet de partager 16 types d'informations, dont, entre autres : profil du patient, visites / rendez vous, allergies et alertes, imagerie médicale, ordonnances, médicaments prescrits, diagnostiques.

Le modèle retenu ici est donc l'opt-in avec restriction. La solution logicielle qui a été choisie est le logiciel HealthShare d'Intersystem [22], société qui s'occupe également du DMP hollandais.

1.2.5 Au Canada

Le gouvernement canadien espère mettre en place un DMP interopérable qui permettrait aux 32 millions canadiens de posséder un dossier électronique gardant trace des antécédents médicaux et permettant un service plus adapté et plus efficace.

De ce fait, une société indépendante à but non lucratif, nommée Inforoute, a été créée en 2001 par les premiers ministres du Canada pour favoriser et accélérer le développement et l'adoption de systèmes de dossier patient partageables (DMP)

21. Diapositives de présentation du LSP d'Intersystems : http://www.intersystems.fr/media/media_manager/pdf/1265.pdf
22. Diapositives de présentation du NPÖ par Intersystems : http://www.intersystems.fr/media/media_manager/pdf/1827.pdf

fondés sur des normes et des technologies de communication compatibles. Financé par le gouvernement du Canada, Inforoute collabore avec les dix provinces et les trois territoires pour mettre en œuvre des systèmes de DMP privés et sécurisés, dont les meilleures pratiques et les projets réussis dans une région peuvent être partagés ou reproduits dans d'autres régions. [23]. Inforoute a pour vocation de mettre en place le DMP pancanadien, méta-DMP qui reposerait sur les DMP de chaque province.

En Ontario et au Québec, le modèle de consentement utilisé est un opt-out : le patient est considéré comme ayant donné son aval pour l'accès à son dossier médical mais peut se retirer des échanges s'il se manifeste.

Les provinces de la Colombie Britannique, Alberta, Manitoba et les provinces atlantiques (Nouveau-Brunswick, la Nouvelle-Écosse, l'île-du-Prince-Édouard et Terre-Neuve-et-Labrador) ont opté pour un modèle sans consentement : les données sont échangées et récoltées sans tenir compte de l'approbation ou du refus du patient.

En Saskatchewan, le patient peut refuser l'accès ou le partage de ses informations mais dans le cadre des soins et des traitements, les informations sont automatiquement échangées.

Ontario

En Ontario il y a déjà des systèmes d'échanges de données cliniques qui ne prennent pas en compte le consentement du patient :

- le SILO, ou système d'information de laboratoire de l'Ontario, récolte les résultats de laboratoires et se charge du partage de ces informations ;
- le PACS, ou système d'archivage et de transmission d'images («*Pitcture Archiving and Communications System*»), récolte et partage les données de type imagerie telles que les radiographies.

Le visualiseur des profils pharmaceutiques des patients, VPPP, est un projet qui permet au Ministère de la santé et des soins longue durée de partager les informations concernant la médication des patients dont les dépenses médicamenteuses constituent une grande part de leur revenus et les personnes âgées (cela fait partie du programme

23. tiré de : https://www.infoway-inforoute.ca/lang-fr/about-infoway

PMO[24] : Programme Médicament de l'Ontario). Concernant le VPPP[25], les informations partagée ne sont que les informations que le patient a consenti à partager. L'accès aux données nécessite une autorisation. L'opt-out n'est pas possible sur l'enregistrement des informations cliniques, mais peut l'être sur la consultation des données. Comme le patient peut masquer certaines données, les praticiens qui consultent les dossiers électroniques sont prévenus que la médication présentée peut être partielle. Le consentement varie selon le type de fournisseur de soins qui accède aux données :

- Pharmacien : consentement implicite, le pharmacien peut accéder aux données du patient sans qu'il y ait besoin que le patient formule expressément son consentement. Il peut consulter les données dans la limite fixée par le patient avec un mot de passe.
- Urgences : consentement implicite ;
- Praticien médical : consentement exprès nécessaire.

Alberta

Le DMP principalement étudié ici en Alberta est l' «Alberta Physician Office System Program[26]» (POSP) qui fait partie d'un projet plus vaste : Alberta Netcare, qui regroupe différents projets administrés par le Ministère de la Santé. Alberta Netcare a pour but de créer un DMP à l'échelle de la province. Le POSP contient les données cliniques et les données de laboratoire du patient ainsi que les informations relatives au patient en vue de son identification. Il est possible pour le patient de choisir quel type de données sera masqué, bien qu'il soit prévu des situations exceptionnelles dans lesquelles l'accès aux données du patient pourrait se faire sans son consentement. Le patient ne peut pas masquer ses informations démographiques.

Colombie britannique

PharmaNet[27] est un DMP qui lie les pharmacies avec les praticiens. Il ne contient que le profil médicamenteux, quelques données cliniques et quelles données démogra-

24. http://www.health.gov.on.ca/french/publicf/pubf/drugsf/odbf.html
25. http://www.ehealthontario.on.ca/FR/programs/dpv.asp
26. http://www.posp.ab.ca/
27. http://www.health.gov.bc.ca/pharmacare/pharmanet/netindex.html

phiques. Il contient également des informations concernant la franchise et la couverture des patients.

Les interactions patient-Pharmanet sont intégralement gérées par la communauté des pharmaciens. Le patient peut protéger ses données avec un mot-clé (fonctionnant comme un mot de passe) afin de masquer ses données, qui sera mis en place après une requête auprès d'un pharmacien.

1.2.6 Québec

Il existe déjà des DMP mis en place au Québec comme le Dossier Santé Québec (DSQ) ou bien le dossier clinique informatisé (DCI) :

- Le dossier clinique informatisé est un dossier de santé qui contient l'ensemble de l'historique médical d'un patient.
- Le dossier santé Québec est un dossier de santé regroupant les informations cliniques de base d'un patient, principalement son profil médicamenteux et ses résultats d'examens de laboratoire et d'imagerie diagnostique.

Dossier Santé Québec

Le Dossier de la Santé du Québec regroupe les données d'immunologie, les profils médicamenteux, les résultats d'imagerie et les résultats de laboratoires. Le DSQ souffre actuellement de plusieurs faiblesses :

- consentement trop grossier : l'utilisateur consent pour toutes ses données sans pouvoir ajuster la portée de son consentement [28] ;
- droits d'accès grossier : la figure 1.1 montre les profils d'accès et leurs droits d'accès associés, soulignant ainsi qu'un médecin a accès l'ensemble des informations du DSQ, même lorsque l'accès à une partie seulement est nécessaire.

Le modèle de consentement utilisé est le consentement implicite afin de« [faciliter la] circulation de l'information entre professionels» selon l'AQESSS [29]. En outre,

28. «Même avec votre consentement, les renseignements de votre DSQ ne peuvent être communiqués à d'autres personnes.», extrait de http://www.dossierdesante.gouv.qc.ca/fr_citoyens_confidentialite.phtml

29. Association québecoise d'établissement de santé et de services sociaux : http://www.aqesss.qc.ca/1777/Communiques_de_presse.aqesss?ComID=631

toutes les données sont obligatoirement ajoutées ou mises à jour dans le DSQ à chaque visite du patient à l'un des professionnels de la santé habilités.

Dossier clinique informatisé

Le DCI, appelé parfois «dossier local», est tenu par un point de service de santé et est donc alimenté par les informations obtenues à ce point de service. Le partage des informations du DCI est limité aux praticiens du point de service de santé au moment de la consultation ou de la période de traitement, mais peut être étendu par le consentement de l'usager à des tiers. Le DCI diffère principalement du DSQ par le fait qu'il ne prend en compte que les données provenant d'un point de service de santé. Le DSQ permet un échange entre les différents points de service de santé où l'usager a reçu des soins. Au niveau du contenu, les deux dossiers diffèrent également : le DCI comprend des informations plus complètes que celles du DSQ parmi lesquelles se trouvent les notes de médecin, diagnostics et antécédents familiaux. Le DCI apparaît comme le complémentaire du DSQ dans le sens où il permet des échanges «locaux» alors que le DSQ permet des échanges «globaux» à l'échelle de la province.

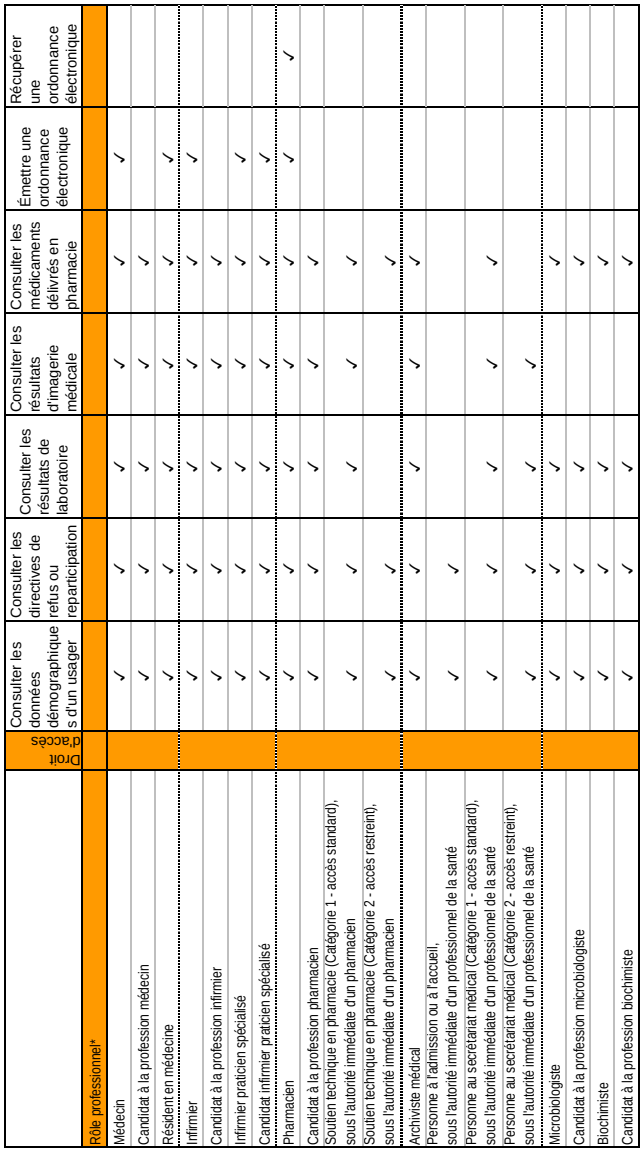
Dossier patient partageable

Le projet de «dossier patient partageable» (DPP) tel que l'envisage la SOGIQUE [30] permettra le partage des différentes données cliniques et médicales déjà existantes dans divers établissements médicaux ou organismes de la santé. Le DPP est donc un DMP qui fonctionne comme une nouvelle base de données qui s'ajouterait à celles déjà existantes : cette base de données ferait le lien entre les bases de chaque établissement en se basant sur un index patient unique, qui n'est pas encore en place car chaque établissement possède pour chaque patient un identifiant unique local. Ainsi un même individu peut avoir plusieurs identifiants différents dans les index patient des établissements hospitalier. C'est pourquoi la SOGIQUE essaye de mettre actuellement en place un index patient maître.

Le consentement proposé par le groupe de travail mandaté par la SOGIQUE se base sur un formulaire électronique sur lequel le patient doit préciser les organisations

30. Société spécialisée en gestion informatique : http://www.sogique.qc.ca/sogique-en-bref.aspx

1.2. Consentement



Rôle professionnel*	Droit d'accès	Consulter les données démographiques d'un usager	Consulter les directives de refus ou repartication	Consulter les résultats de laboratoire	Consulter les résultats d'imagerie médicale	Consulter les médicaments délivrés en pharmacie	Émettre une ordonnance électronique	Récupérer une ordonnance électronique
Médecin		✓	✓	✓	✓	✓	✓	
Candidat à la profession médecin		✓	✓	✓	✓	✓		
Résident en médecine		✓	✓	✓	✓	✓	✓	
Infirmier		✓	✓	✓	✓	✓	✓	
Candidat à la profession infirmier		✓	✓	✓	✓	✓		
Infirmier praticien spécialisé		✓	✓	✓	✓	✓	✓	
Candidat infirmier praticien spécialisé		✓	✓	✓	✓	✓	✓	
Pharmacien		✓	✓	✓	✓	✓	✓	✓
Candidat à la profession pharmacien		✓	✓	✓	✓	✓		
Soutien technique en pharmacie (Catégorie 1 - accès standard), sous l'autorité immédiate d'un pharmacien		✓	✓	✓	✓	✓		
Soutien technique en pharmacie (Catégorie 2 - accès restreint), sous l'autorité immédiate d'un pharmacien		✓	✓			✓		
Archiviste médical		✓	✓	✓	✓	✓		
Personne à l'admission ou à l'accueil, sous l'autorité immédiate d'un professionnel de la santé		✓	✓					
Personne au secrétariat médical (Catégorie 1 - accès standard), sous l'autorité immédiate d'un professionnel de la santé		✓	✓	✓	✓	✓		
Personne au secrétariat médical (Catégorie 2 - accès restreint), sous l'autorité immédiate d'un professionnel de la santé		✓	✓	✓	✓			
Microbiologiste		✓	✓	✓		✓		
Candidat à la profession microbiologiste		✓	✓	✓		✓		
Biochimiste		✓	✓	✓		✓		
Candidat à la profession biochimiste		✓	✓	✓		✓		

* Le masculin est utilisé pour alléger le texte.

Figure 1.1 – Droits d'accès des différents profils dans le DSQ
tiré de http://www.dossierdesante.gouv.qc.ca/

de services qui peuvent visualiser le contenu du DPP et également l'alimenter. Mais les responsables du projet préfèrent utiliser un modèle simplifié du consentement qui s'apparente à un consentement comme celui du DSQ : l'usager autorise par écrit tous les utilisateurs du réseau à accéder à distance à des éléments du dossier selon des droits d'accès différenciés.

1.2.7 Synthèse

Le tableau 1.1 donne une synthèse des différents modes de consentement sur la possibilité de se retirer de la collecte des données, de ne pas partager les informations, ou encore de choisir quelles données partager et à qui.

Mécanisme de consentement	Retrait de la collecte de données possible	Partage des données	Filtrage par rapport aux données	Filtrage par rapport à l'entité
Canada				
Alberta POSP		opt-out	×	×
Colombie Britannique		opt-out	×	
Ontario VPPP		opt-out	×	
Québec DSQ	×	opt-out		
États-Unis				
Delaware		opt-out		
Indiana		opt-out		
Maryland		opt-out	× [31]	
New York		opt-in		×
Massachussetts		opt-in		×
Rhode Island		opt-in		×
Washington	×	opt-in	×	×
Europe				
Suède		opt-in		×
Pays-bas	×	opt-out	×	×
Royaume-Uni		opt-out	×	

Tableau 1.1 – Synthèse des DMP existants

31. suivant les professionnels de la santé : le CRISP laisse la charge/responsabilité de la prise en charge du consentement plus fin aux organisations fournissant les soins.

1.3 Gestion du consentement au CHUS

Cette section a pour but de présenter le projet mis en place avec le CHUS. Il permettra à terme d'échanger de manière simple, sécurisée et avec l'aval du patient les informations entre le CHUS et les centres de santé et de services sociaux. Ce projet s'appuiera sur ARIANE et la réalisation d'un prototype pourra peut-être faire avancer le DSQ et à terme, être réutilisé dans le DSQ. La figure 1.2 présente le projet d'interface des différentes institutions médicales du Québec. Le but est d'obtenir un DMP plus avancé que le DSQ qui prend en compte un consentement plus granulaire que le consentement du DSQ.

1.3.1 Cas étudiés

Au fil de nos différentes entrevues avec le personnel du CHUS, plusieurs cas d'utilisation du consentement gérés au sein de l'établissement ont été exprimés. Ces cas définissent les différents protocoles hospitaliers avec la gestion du consentement associée. Sur les cas étudiés, on suppose que les droits d'accès sont fournis suivant le profil d'accès du requérant, pratique déjà en place au CHUS. En effet, une infirmière chef de service n'aura pas les mêmes droits d'accès que les apprenties infirmières.

Transfert de patient

Dans le cas d'un transfert de patient, il n'est exigé aucun consentement écrit car le consentement est implicite : le patient consent à se faire transférer et à partager ses données avec l'établissement d'arrivée. Les données accessibles par l'établissement d'arrivée sont limitées à l'épisode en cours. Si besoin il y a, l'établissement pourra demander au patient son consentement pour accéder à des données supplémentaires. La durée de validité du droit d'accès peut être prolongée lorsque les résultats des tests sont en attente de révision par exemple.

- Le transfert de service : le CHUS délègue une tâche «simple» à un autre établissement, par exemple un CLSC ou un centre d'accueil qui pourront assurer un suivi du patient à proximité de celui-ci.
- Le transfert de référence : le CHUS réfère le patient dans un même service mais appartenant à un autre centre, par exemple lorsque le CHUS ne possède pas

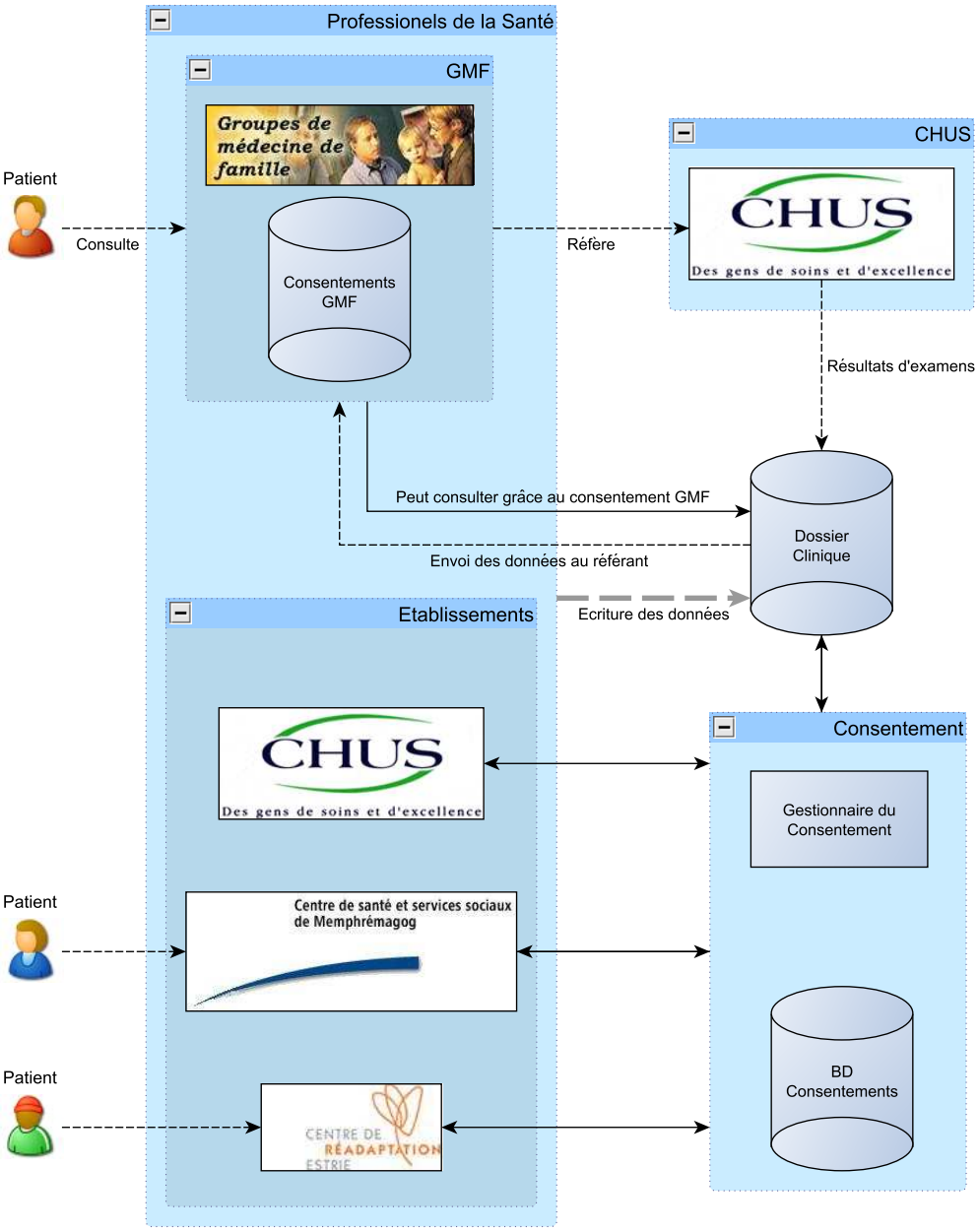


Figure 1.2 – Projet : Interface des différentes institutions

l'infrastructure nécessaire.

– Le transfert de centre hospitalier : le CHUS transfert un patient vers un autre centre hospitalier, par exemple pour le rapprocher de son domicile.

Transfert dans un CRE

Le consentement écrit du patient pour ce transfert est nécessaire car il ne s'agit pas d'un transfert entre centres hospitaliers. Les CRE doivent pouvoir consulter le dossier du patient avant de pouvoir l'accepter, pour voir s'ils ont la capacité d'accueillir le patient.

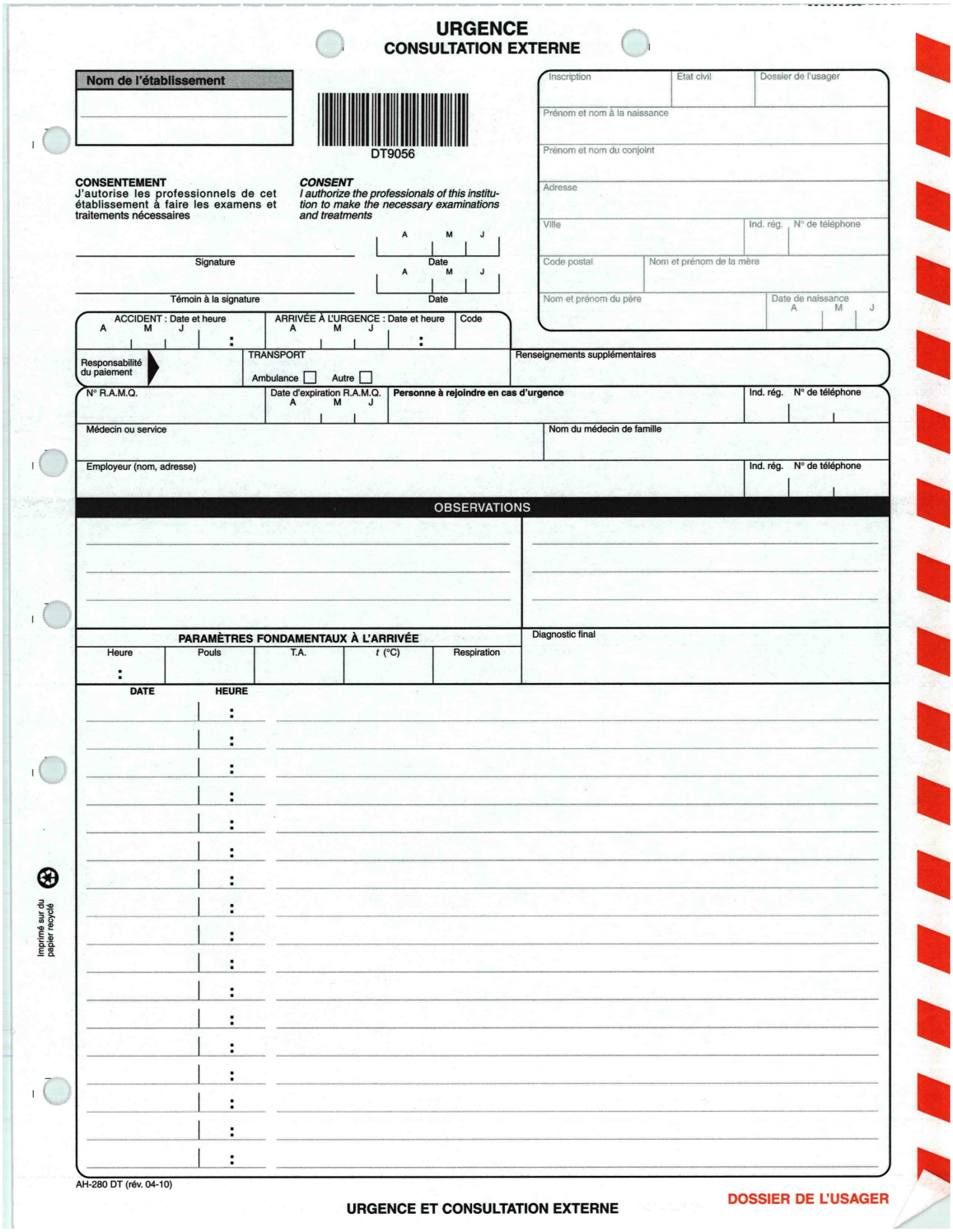
Médecin pratiquant en clinique privée

Pour toute demande d'accès, le patient doit autoriser la transmission d'information à un médecin via une autorisation suivant la loi. Une pratique interne au CHUS est de regarder dans le dossier patient, lors d'une demande d'un médecin (sans autorisation du patient), si le nom du médecin apparait dans les données du patients datées de moins d'un an, ce qui permet de savoir s'il est suivi par ce médecin. S'il s'avère que le médecin n'est pas dans le dossier médical, il faut le consentement écrit du patient. Dans le cas d'un médecin GMF (groupe de médecine familiale), si le patient a signé le consentement GMF (d'une durée d'un an renouvelable stipulant que n'importe quel médecin GMF peut avoir accès à son dossier), et si le médecin appartient au groupe GMF auquel le patient a donné son consentement, la totalité du dossier du patient est alors accessible au médecin via ARIANE. Dans le cas d'un médecin prescripteur, un médecin qui réfère un patient au CHUS, les données de l'examen lui appartiennent, le consentement du patient étant implicitement donné lorsqu'il vient passer les examens au CHUS. Les données des examens sont donc automatiquement envoyées au médecin prescripteur sans consentement.

Envoi des AH-280 (notes d'urgences)

Sur la note d'urgence, voir sur la figure 1.3, survient un cadre où le consentement apparaît : si le patient reçoit des soins il consent à ce que les intervenants aient accès à son dossier. De plus, un cadre spécifique pour le médecin de famille est présent : le patient choisit s'il désire que les informations issues de l'intervention soient transmises au médecin de famille ou non. 80% des AH-280 n'ont pas de noms inscrit dans le cadre réservé au médecin de famille.

Cependant, l'ensemble des données issues de cette visite, appelé épisode, en lien avec l'AH-280 ne devrait pas être accessible au médecin GMF si le patient a refusé



URGENCE
CONSULTATION EXTERNE

Nom de l'établissement

DT9056

Inscription | Etat civil | Dossier de l'usager

Prénom et nom à la naissance

Prénom et nom du conjoint

Adresse

Ville | Ind. rég. | N° de téléphone

Code postal | Nom et prénom de la mère

Nom et prénom du père | Date de naissance A M J

CONSENTEMENT
J'autorise les professionnels de cet établissement à faire les examens et traitements nécessaires

CONSENT
I authorize the professionals of this institution to make the necessary examinations and treatments

Signature | Date A M J

Témoin à la signature | Date A M J

ACCIDENT : Date et heure A M J : | ARRIVÉE À L'URGENCE : Date et heure A M J : | Code

Responsabilité du paiement | TRANSPORT Ambulance ☐ Autre ☐ | Renseignements supplémentaires

N° R.A.M.Q. | Date d'expiration R.A.M.Q. A M J | Personne à rejoindre en cas d'urgence | Ind. rég. N° de téléphone

Médecin ou service | Nom du médecin de famille

Employeur (nom, adresse) | Ind. rég. N° de téléphone

OBSERVATIONS

PARAMÈTRES FONDAMENTAUX À L'ARRIVÉE

Heure	Pouls	T.A.	t (°C)	Respiration
:				

Diagnostic final

DATE	HEURE	
	:	
	:	
	:	
	:	
	:	
	:	
	:	
	:	
	:	
	:	
	:	
	:	
	:	
	:	
	:	
	:	

Imprimé sur du papier recyclé

AH-280 DT (rév. 04-10)

URGENCE ET CONSULTATION EXTERNE

DOSSIER DE L'USAGER

Figure 1.3 – Formulaire AH-280

explicitement (signé un refus) de transmettre l'information. Il est très rare que le patient signe un refus d'envoi d'un document. Si le patient n'a pas indiqué le nom de son médecin de famille mais que celui-ci le demande, il a le droit de le faire étant donné que le consentement GMF le lui permet. En temps normal, le médecin demande de lui envoyer la AH-280 lorsqu'il est informé que le patient a été vu au CHUS. Habituellement, le médecin reçoit cette information directement du patient.

Patient décédé
Dans le cas d'un patient décédé, il faut fermer tous les accès aux données du patient sauf pour des cas particuliers (exceptions prévues par la loi). Cela restera géré par les archivistes.

Délégation
Dans le cas où le patient serait dans l'incapacité d'émettre un consentement libre et éclairé, il y a la possibilité qu'un représentant légal puisse émettre un consentement à sa place : par exemple si le patient est un enfant trop jeune, ou bien si le patient est dans le coma.

Ententes avec des établissements tels l'IUGS ou l'hémodialyse de Magog
Dans le cas des établissements avec des ententes, le dossier médical du patient est dans le CHUS. Le médecin a présentement accès à la totalité du dossier médical. Si le patient signe un consentement, alors le médecin ne devrait avoir accès que s'il est le médecin traitant du patient.

1.3.2 ARIANE

ARIANE est une application du CHUS qui comprend des modules visant à centraliser les informations du patient au sein de l'établissement, à savoir :

- index patient : index qui référence le patient de manière unique au sein de l'établissement ;
- ADT (admission, départ, transfert) : module qui gère tous les transferts et tous les événements du patient ;

- module de gestion des laboratoires : module qui gère les résultats des laboratoires ;
- module d'imagerie : module qui gère les résultats des examens d'imagerie, à savoir
  - résonance magnétique ;
  - électrophysiologie ;
  - radiographie ;
  - examens neurologiques ;
- module qui stocke les signes vitaux ;
- module qui stocke les évaluations faites à l'arrivée du patient et pendant son séjour ;
- module du médecin : module qui garde tous les résultats, les ordonnances, les prescriptions, et les demandes d'examens ;
- module qui gère les rendez-vous.

De plus cette application permet d'avoir en visuel tous les documents numérisés, y compris les documents manuscrits des médecins. Cette application a été développée par QuadraMed, sous le nom QCPR (« *QuadraMed Computerized Patient Record* »).

1.3.3 Graphe conceptuel d'ARIANE

Un graphe conceptuel du fonctionnement d'ARIANE, présenté à la figure 1.4, a été établi auprès de l'équipe du CHUS.

Sécurité

Dans le cadre d'ARIANE, une sécurité correspond à un rôle en RBAC. Chaque sécurité n'est définie que pour un établissement donné.

Chart Review

Le Chart Review désigne le sous-ensemble des colonnes accessibles parmi les informations affichées à l'écran. C'est donc un moyen de filtrer des données partiellement en masquant certains attributs dans la base de données. Le Chart Review masque seulement les colonnes (donc les attributs des bases de données) mais en aucun cas ne masque les lignes (enregistrements). C'est actuellement le seul moyen de masquer des informations des patients et est donc incomplet : par

1.3. Gestion du consentement au CHUS

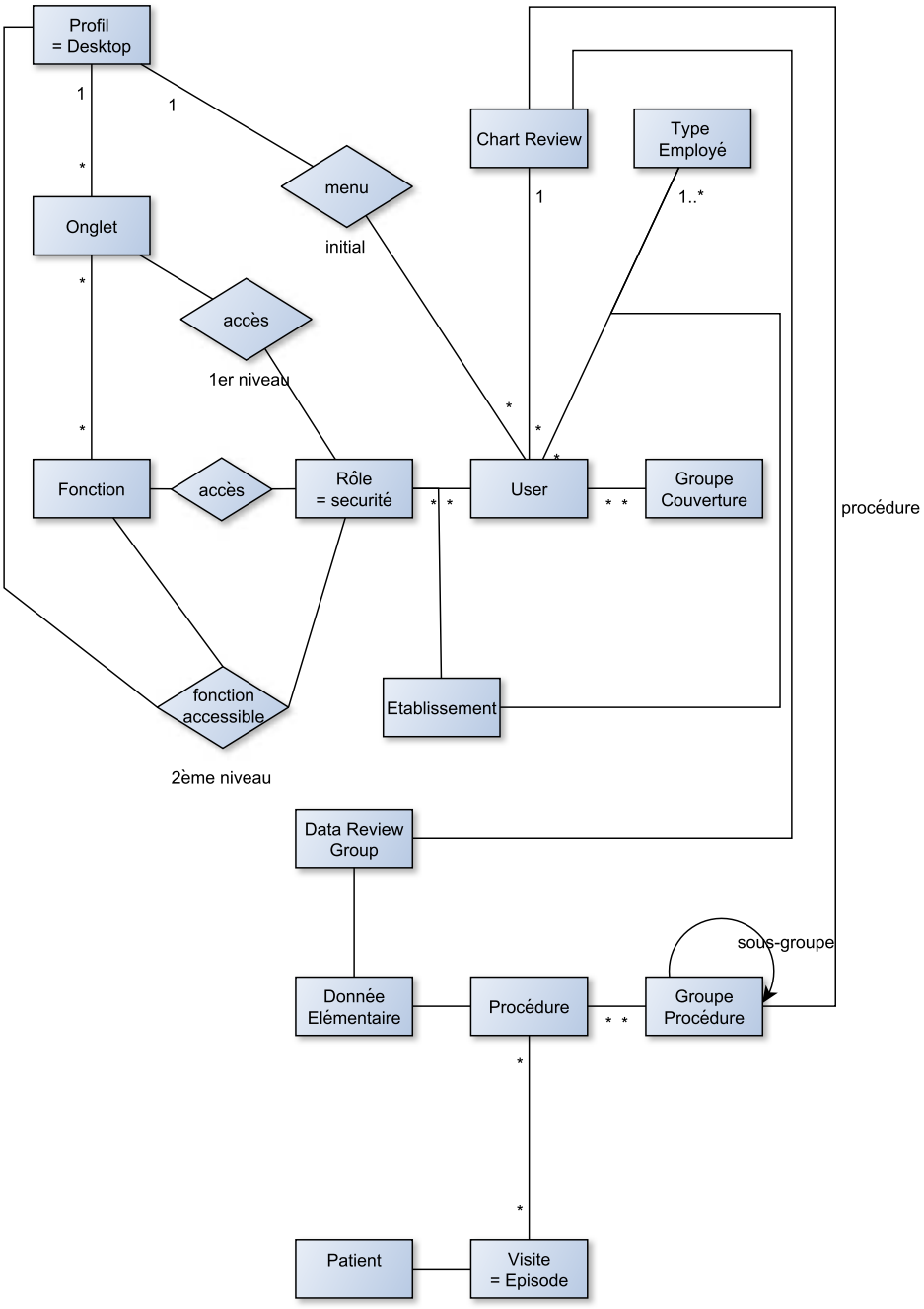


Figure 1.4 – Concepts d'ARIANE

exemple, on ne peut empêcher un médecin du CHUS de savoir que son patient actuel a consulté à Magog.

Profil

Un profil ou encore desktop, est l'ensemble des fonctions accessibles pour l'intervenant. Ces fonctions sont regroupées par onglets.

Groupe de couverture

C'est le groupe des collègues susceptibles de remplacer le médecin, récupérant les patients de celui-ci lorsqu'il est absent. Par contre, que le médecin à couvrir soit absent ou non, il est toujours possible pour un médecin du groupe de couverture d'accéder en lecture au dossier d'un des patients du médecin couvert.

Procédure

Une procédure est constituée des données élémentaires accumulées lors d'une procédure médicale qui peut s'étendre sur plusieurs épisodes. Ainsi des prises de sangs répétées dans le cadre d'un dépistage font toutes partie de la procédure de dépistage. Il faut toujours passer par une procédure (via un épisode) pour accéder à des données. Actuellement, ARIANE propose au médecin une sélection personnalisée de procédures mais rien n'empêche le médecin de regarder d'autres procédures : les médecins ont accès à toutes les procédures.

1.3.4 Structure de la base de données d'ARIANE

N'ayant pas accès à l'implémentation d'ARIANE, un travail de rétro-ingénierie est obligatoire afin de savoir sous quelle forme sont stockées les informations, et quelles sont les informations disponibles dont va pouvoir disposer notre prototype. La figure 1.5 résume le modèle de base de données utilisée par ARIANE esquissé au fil de nos entrevues avec le CHUS.

1.3. Gestion du consentement au CHUS

Figure 1.5 – Structure de la base de données du CHUS

1.4 Solutions existantes

Cette section s'intéresse aux solutions ou concepts qui permettent de modéliser le consentement. Dans la littérature, on retrouve divers modèles de DMP centrés sur le patient tels que les modèles de Coiera *et al*[3], de Bergmann *et al* [1], de Huda *et al* [16] ou de Jin *et al* [12]. La majorité de ces modèles ne présentent pas un modèle de consentement assez précis. Seuls les modèles de Coiera *et al* [3], de Russello *et al* et XACML sont présentés.

1.4.1 Un exemple d'outil modélisant le consentement : e-Co

Le modèle e-Co (pour *e-consentement*) s'appuie sur 4 des 5 modes de consentements présentés à la section 1.2 : **opt-in complet, opt-out complet, opt-in avec restrictions, opt-out avec restrictions**.

Il faut noter que l'*opt-in avec restrictions* introduit dans [3] correspond à l'*opt-out partiel* définit dans 1.2 et l'*opt-out avec restrictions* correspond à l'*opt-in partiel*.

L'idée générale est que le consentement pourrait être représenté par une liste de règles. Ces règles sont de la forme :

```
ACCESS TO <INFORMATION>
BY AN <ENTITY>
FOR A <PURPOSE>
IN A <CONTEXT>
IS {CONSENTED TO | DENIED}.
```

`ENTITY` pourrait être un rôle comme en RBAC, et `INFORMATION` pourrait être une portion de dossier. Par exemple, si un patient consent à ce qu'un chirurgien ait accès aux radiographies de ses poumons dans le cadre d'un diagnostique préopératoire, son consentement serait de la forme suivante :

```
ACCESS TO <LUNGS RADIOGRAPHY>
BY A <SURGEON>
FOR AN <OPERATION>
IN A <PREOPERATIVE CONTEXT>
IS {CONSENTED TO}.
```

A partir de ces consentements, il est possible de les caractériser en deux types : des interdictions et des permissions. Il n'est pas possible de mettre dans une même liste des interdictions et des permissions car cela engendre des conflits. Afin de prendre en compte les deux types de consentement, il est nécessaire de faire 2 listes distinctes : les listes de permissions (à la manière des listes blanches) appelée «*in-list*» et les listes d'interdiction (listes noires) appelée «*out-list*». A partir de là, il est possible de modéliser les quatre modes de consentement pris en compte : il suffit de prendre en compte une liste, voire les deux, et dans le dernier cas définir une priorité entre les listes.

Ainsi dans le cas d'un *opt-in complet*, la liste noire est vide et la liste blanche contient tous les médecins affectés par la portée du consentement. La figure 1.6 montre comment sont organisées la liste blanche et la liste noire dans le cas d'un *opt-in complet* permettant à tous les médecins du CHUS d'accéder au dossier.

Dans le cas d'un opt-in avec restriction ou d'un opt-out avec restriction, il faut gérer la priorité entre les deux listes. Dans le cas d'un *opt-in avec restrictions*, les restrictions doivent avoir préséance sur les autorisations accordées, la liste noire est donc prioritaire. Dans le cas d'un *opt-out avec restrictions*, c'est l'inverse. La figure 1.7 illustre le cas d'un patient souhaitant permettre à tous les intervenants d'avoir accès à son dossier, sauf les infirmières qui ne peuvent avoir accès à ses données psychiatriques.

```
[in-list]
ACCESS TO <*>
BY AN <MEDECIN DU CHUS>
FOR A <*>
IN A <*>
IS CONSENTED TO

[out-list]
null
```

Figure 1.6 – in-list et out-list pour un opt-in global consenti aux médecin du CHUS.

E-Co possède donc une méthode de résolution de conflit **statique** non adaptée à nos besoins.

```
[in-list]
ACCESS TO <*>
BY AN <INTERVENANT DU CHUS>
FOR A <*>
IN A <*>
IS CONSENTED TO

[out-list]
ACCESS TO <DONNEES PSYCHIATRIQUES>
BY AN <INFIRMIERE>
FOR A <*>
IN A <*>
IS DENIED
```

Figure 1.7 – in-list et out-list pour un opt-in avec restrictions empêchant les infirmières d'avoir accès aux données psychiatriques.

1.4.2 XACML

XACML est l'acronyme de eXtensible Access Control Markup Language. XACML est un langage de spécification de politiques (ensemble de règles) d'accès. C'est un standard OASIS (Organization for the Advancement of Structured Information Standards)[32] La structure d'XACML s'appuie sur la présence d'un PEP («*policy enforcement point*»), qui garde l'accès aux ressources à protéger, et d'un PDP («*policy decision point*») qui est chargé de prendre les décisions en fonction des différents paramètres qui peuvent lui être fournis par le PIP («*policy information point*»). La figure 1.8 illustre la structure de XACML : lorsqu'on effectue une requête d'accès à des ressources protégées, le PEP interroge le PDP avec des informations telles que l'identité de l'auteur de la requête, l'action demandée, les ressources désirées et des paramètres de l'environnement. Le rôle du PDP est d'évaluer la requête en fonction des règles et des différents paramètres fournis par le PIP. A partir de là, le PDP peut renvoyer au PEP 4 types de réponses :

1. Accès accordé ;
2. Accès interdit ;

32. http://www.oasis-open.org/

3. Indétermination : une erreur s'est produite, où des données sont manquantes empêchant le PDP de prendre une décision ;
4. Non applicable : la requête ne peut être évaluée par le PDP.

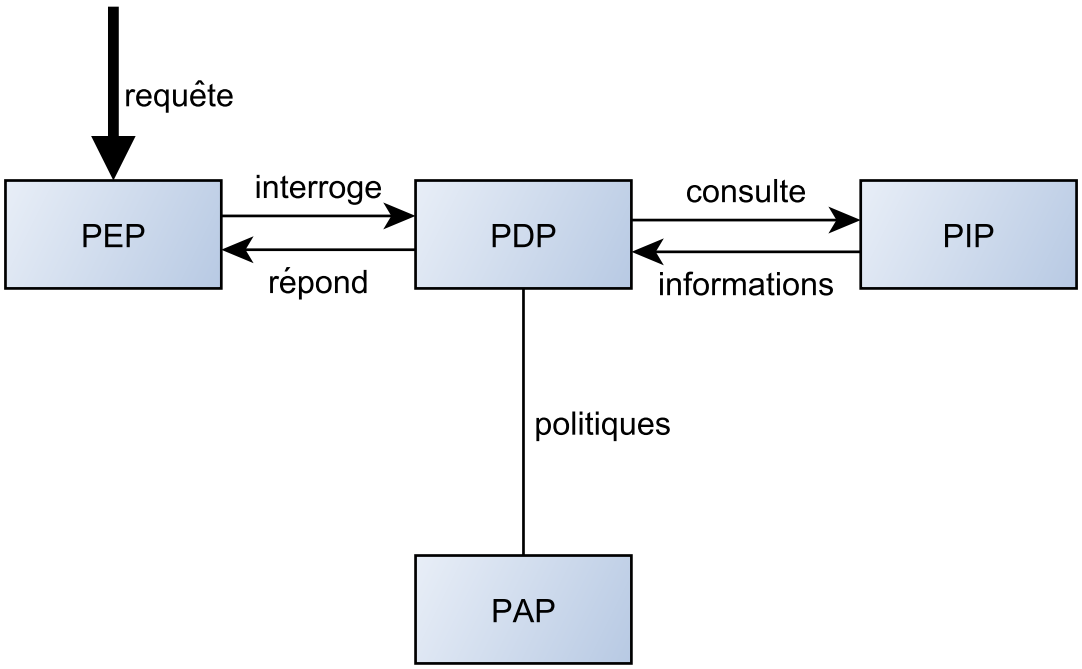


Figure 1.8 – Architecture de XACML

Structure d'une règle XACML

Une politique d'accès est composée de règles. Chaque règle est constituée de :

- une cible : elle-même constituée d'attributs tels que
  - sujets : entités à l'origine de la requête, qui souhaite exécuter une ou plusieurs actions sur les ressources ;
  - ressources : les informations auxquelles les sujets désirent accéder ;

  - actions : actions que les sujets veulent exécuter sur les ressources ;
  - environnement ;
- un effet : permission ou interdiction ;
- une condition d'application ;

La condition des règles de la politique permet de renvoyer les différents résultats lorsqu'une politique applicable est trouvée : une condition est une fonction booléenne qui, lorsqu'elle est évaluée à VRAI, permet de renvoyer l'effet de la règle (permission ou interdiction). Lorsque cette fonction est évaluée à FAUX, le résultat «non applicable» est retourné alors que si une erreur surgit, le résultat est l'indétermination.

Gestion des conflits modaux

Un conflit modal est un conflit entre deux règles de modalité différente, à savoir une autorisation et une interdiction. XACML possède un système de gestion de conflits modaux appelés «Combining Algorithms». XACML contient déjà des «Combining Algorithms» basiques :

- «deny overrides» : si une règle interdit l'accès dans la politique, alors la requête est déboutée ;
- «ordered deny overrides» : se distingue d'un «deny overrides» par l'ordre dans lequel les règles sont évaluées, qui est le même que l'ordre dans lequel les règles ont été ajoutées ;
- «permit overrides» : si une règle autorise l'accès dans la politique, alors la requête est accordée ;
- «ordered permit overrides» : se distingue d'un «permit overrides»par l'ordre dans lequel les règles sont évaluées, qui est le même que l'ordre dans lequel les règles ont été ajoutées ;
- «first applicable» : la première règle applicable définit le comportement de la politique ;

Bien qu'il soit possible d'ajouter des «Combining Alorithms», la procédure semble compliquée puisqu'il faut spécifier le comportement en dur suivant chaque cas.

Voici un exemple d'implémentation de «permit overrides» en Java :

programme 1.1 – Implémentation de permit-overrides

```
public class PermitOverridesRuleAlg extends RuleCombiningAlgorithm {
```

```
    public PermitOverridesAlg() throws URISyntaxException {
        super(new URI("rule-combining-alg:permitOverrides"));
    }

    public Result combine(EvaluationCtx context, List rules) {
        Iterator it = rules.iterator();
     Boolean flag = false;

        while (it.hasNext()) {
            // get the next Rule, and evaluate it
            Rule rule = (Rule)(it.next());
            Result result = rule.evaluate(context);

            // if it returns Permit, then the alg returns Permit
            if (result.getDecision() == Result.DECISION_PERMIT)
                return result;

           //if it returns Deny, then set the flag to TRUE
           if ((!flag)&(result.getDecision() == Result.DECISION_DENY))
                flag = true;
        }
    //if nothing returned Permit but Deny appeared, then the alg returns Deny
    if(flag)
    return new Result(Result.DENY);

        // if nothing returned Permit or Deny, then the alg returns Not Applicable
        return new Result(Result.NOT_APPLICABLE);
    }

}
```

Travaux connexes

L'un des principaux avantages de XACML est le fait qu'il soit supporté par une communauté grandissante. Les recherches de Jeremy Bryans[2] montrent qu'il est possible de représenter la base du langage XACML en CSP[11](Communicating Sequential Processes) qui est une algèbre de processus. Ainsi il est possible de spécifier des propriétés des politiques XACML en CSP et vérifier que les politiques respectent bien ces propriétés.

D'autres travaux menés par Zhang*et al* [20]montrent qu'il est possible de générer des politiques en XACML à partir d'un langage proche de Prolog, le langage *RW*.

Ces travaux sont intéressants pour notre projet dans la mesure où il faudrait que :

- on puisse avec notre solution faire de la vérification afin de pouvoir vérifier par exemple qu'un patient pourra toujours se faire soigner dans un établissement ;
- on puisse générer les règles et politiques à partir d'un langage déclaratif tel que Prolog, pour faciliter la tâche des analystes de sécurité ; une approche déclarative permettant d'ajouter, de modifier ou supprimer des règles aisément.

Malgré cela, XACML ne possède pas que des attraits :

- il est difficile de représenter une hiérarchie de sujets et de ressources ;
- par défaut, la gestion des priorités entre les règles est presque inexistante, sauf selon l'ordre de déclaration ;
- la syntaxe de XACML est très verbeuse et peu lisible.

Une autre interface pour créer les règles, plus simple serait un grand atout.

1.4.3 Ponder

Comme dit précédemment, le choix du langage est crucial. Ponder de l'Imperial College London [5, 18, 19] répond aux exigences imposées par l'utilisation de règles dites «dynamiques» : les règles ne doivent pas être statiques et peuvent être ajoutées, modifiées ou bien supprimées facilement. Ponder est un langage déclaratif, très fortement orienté objet.

Ponder introduit la notion de sujet et de cible entre lesquels se placent plusieurs PEP :

- le sujet est l'acteur ;
- la cible est la cible de l'action effectuée par le sujet.

Dans cette configuration sujet/cible représentée à la figure 1.9, on peut distinguer 4 points :

1. requête sortante du sujet ;
2. requête entrante au niveau de la cible ;
3. réponse sortante au niveau de la cible ;
4. réponse entrante au niveau du sujet.

C'est au niveau de ces 4 points cruciaux que l'on va placer des PEP. La figure 1.9 représente un exemple d'utilisation des 4 PEP.

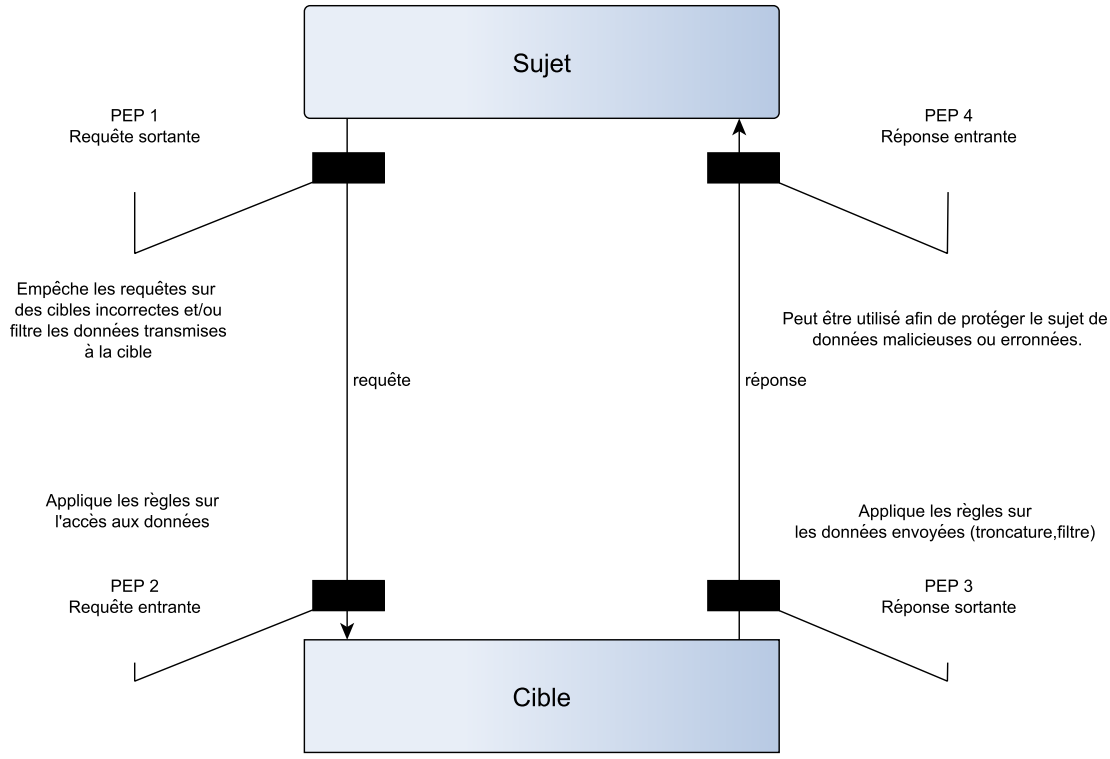


Figure 1.9 – Exemple d'utilisation des PEP

Dans ce langage, les entités qui consomment l'information (sujets) sont placées dans une hiérarchie arborescente appelée « hiérarchie de domaine » aux côtés des entités qui fournissent l'information (cibles). Les sujets partagent la même hiérarchie que les cibles.

Ponder, en outre, possède un système de gestion de conflit autonome permettant de résoudre des conflits modaux, en se basant uniquement sur les règles et ne nécessitant pas l'implémentation directe du comportement du système en cas de conflit comme en XACML.

1.5 Synthèse

Cinq modèles de consentement ont été répertoriés, l'opt-out étant le modèle le plus largement utilisé car permettant un accès par défaut aux données du patient, ce qui facilite le travail des intervenants. Mais parmi les modèles étudiés, aucun ne permet d'avoir une granularité assez fine afin qui correspondrait aux contraintes imposées par les cas du CHUS. Des solutions ont été étudiées dont e-Co, XACML et Ponder. e-Co gère de manière statique les conflits – ce qui rend cette solution inappropriée – et XACML nécessite une implémentation des «combining algorithms», ce qui rend la tâche ardue. Ponder semble donc être le meilleur candidat pour une évaluation plus poussée.

Chapitre 2

Évaluation de Ponder

Ponder possède un système de gestion de conflit autonome par opposition à XACML, ce qui lui permet d'être potentiellement plus adapté à la problématique. Dans ce chapitre, nous nous intéressons au fonctionnement de Ponder pour comprendre comment sont gérés les conflits modaux. Les concepts fondamentaux tels que la hiérarchie de domaine et la structure des règles seront définis dans un premier temps.

2.1 Hiérarchie de domaine

Une hiérarchie de domaines peut être considérée comme un graphe orienté acyclique constitué de domaines et d'objets. Un domaine ne peut avoir qu'un prédécesseur mais peut avoir plusieurs successeurs, domaines ou objets. Un objet ne peut avoir aucun successeur, mais peut avoir plusieurs prédécesseurs. Ainsi chaque domaine est identifié de manière unique par le chemin qui le relie à la racine. On pourrait faire l'analogie avec l'arborescence des fichiers : un dossier est identifié par son chemin absolu. Par contre, un unique fichier dans le cas d'une hiérarchie de domaine pourrait appartenir à plusieurs dossiers. La hiérarchie de domaine permet d'utiliser la notion d'héritage : une règle affectant un domaine, affecte les successeurs du domaine. Ainsi sur la figure 2.1, les domaines sont représentés par des ellipses alors que les objets sont représentés par des rectangles. Les règles affectant le domaine «atteint de la grippe» concernent également le patient A et le patient B.

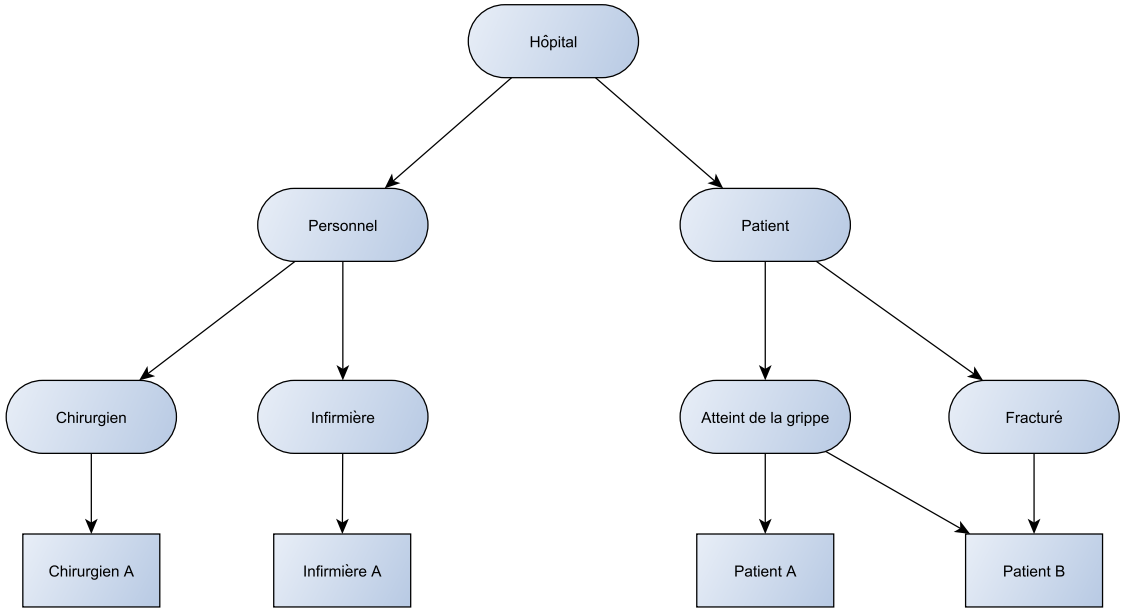


Figure 2.1 – Exemple de hiérarchie de domaines

2.2 Structure d'une règle

Une règle en Ponder est définie par :

- modalité : le type de règle (autorisation ou interdiction) ;
- sujet : entité qui demande la permission d'effectuer l'action ;
- cible : entité cible de l'action ;
- action : action que veut effectuer le sujet sur la cible ;
- clause : prédicat dépendant du sujet, de la cible et des paramètres de l'action qui permet de spécifier une condition d'application de la règle.

Voici un exemple de règle :

$$\overbrace{\text{interdiction}}^{modalité}\ \overbrace{\text{à un médecin X}}^{sujet}\ \overbrace{\text{d'accéder}}^{action}\ \overbrace{\text{au dossier du patient P}}^{cible}$$

$$\underbrace{\text{lorsque X n'est pas le médecin traitant de P}}_{clause}$$

En Ponder, chaque sujet ou cible d'une règle doit être présent dans la hiérarchie de domaine. Par exemple dans la figure 2.1, il faudrait que le dossier du patient soit dans la hiérarchie de domaine en tant qu'objet à la place du patient. Par abus, on confondra le patient et son dossier médical.

2.3 Modalité par défaut

Il est nécessaire de savoir que faire lorsqu'une action est demandée par un sujet et qu'il n'y a aucune règle applicable pour le triplet <sujet,cible,action>. C'est pour cela qu'il faut définir un choix par défaut, la *modalité par défaut* dont voici les deux possibilités.

- Par défaut toute action est autorisée : tant qu'une action n'est pas interdite, elle est autorisée.
- Par défaut toute action est interdite : tant qu'ne action n'est pas autorisée, elle est interdite ;

Ce choix est un attribut d'un domaine : en effet il est possible que dans un établissement donné, toute action est par défaut interdite alors que dans un autre elles sont autorisées. Par hérédité, cet attribut est transmis à tous les éléments fils du domaine et ne peut être donc utilisé qu'à la racine pour des raisons de conflits : on ne pourrait décider dans le cas d'un objet qui appartiendrait à deux domaines disjoints ayant un attribut différent de quel domaine parent l'objet hériterait.

2.4 Stratégie de résolution de conflits

L'utilisation d'une hiérarchie de domaine introduit inexorablement la notion de conflit dont un exemple est représenté à la figure 2.2. Sur cette figure, le chirurgien X voulant accéder au dossier du patient Y est confronté à un problème de conflit : d'une part, son appartenance au domaine Personnel Medical lui interdit d'accéder au dossier des entités du domaine Patients, d'autre part étant chirurgien, il bénéficie

d'une autorisation de consulter les dossier des patients de chirurgie dont fait partie le patient Y.

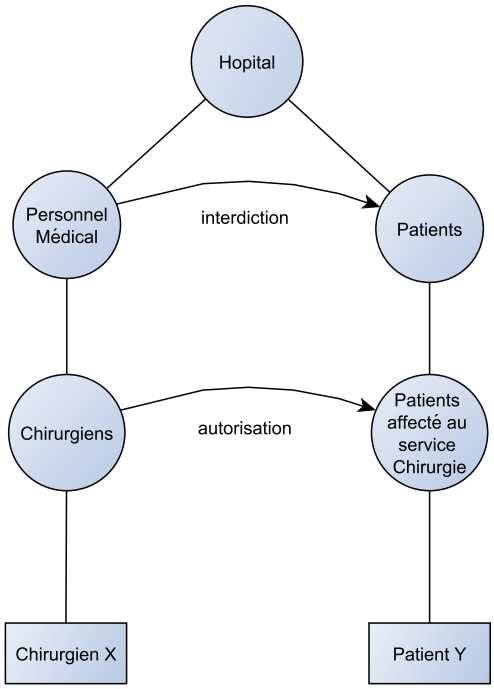


Figure 2.2 – Exemple de conflit simple

En Ponder, il est possible de marquer des règles étant plus prioritaires que d'autres : ces règles sont dites **finales**. Le fait qu'une règle soit finale lui donne précédence sur une règle non finale. Mais cela reste insuffisant pour régler les différents conflits qui peuvent survenir au sein de règles finales.

Il est donc nécessaire d'utiliser un mécanisme de résolution de conflit : l'algorithme 1 est celui utilisé par Ponder pour résoudre un conflit. Une règle est plus basse qu'une autre si son sujet est un descendant du sujet de l'autre règle, et si elles ont le même sujet, la profondeur de la cible est prise en compte. Deux règles peuvent ne pas être comparables comme par exemple à la figure 2.3 sur le graphe (1).

Pour illustrer l'algorithme 1, la figure 2.3 montre les différents types de conflit qu'il peut y avoir :

- si p1 et p2 sont **finales**, alors p1 a précédence sur p2 dans (2) et (3) ;

Algorithme 1 Algorithme de gestion de conflit de Ponder

Entrées : <sujet,cible,action,clause>
Sorties : permission ou interdiction

```
Rechercher les règles applicables au quadruplet <sujet,cible,action,clause>
si aucune règle n'est trouvée alors
    retourner modalité par défaut
finsi
si des règles applicables sont finales alors
    Prendre les règles finales les plus élevées au niveau de la hiérarchie
    si un conflit modal apparait alors
        retourner interdiction
    sinon
        retourner modalité des règles trouvées
    finsi
sinon
    Prendre les règles les plus basses au niveau de la hiérarchie
    si un conflit modal apparait alors
        retourner interdiction
    finsi
finsi
```

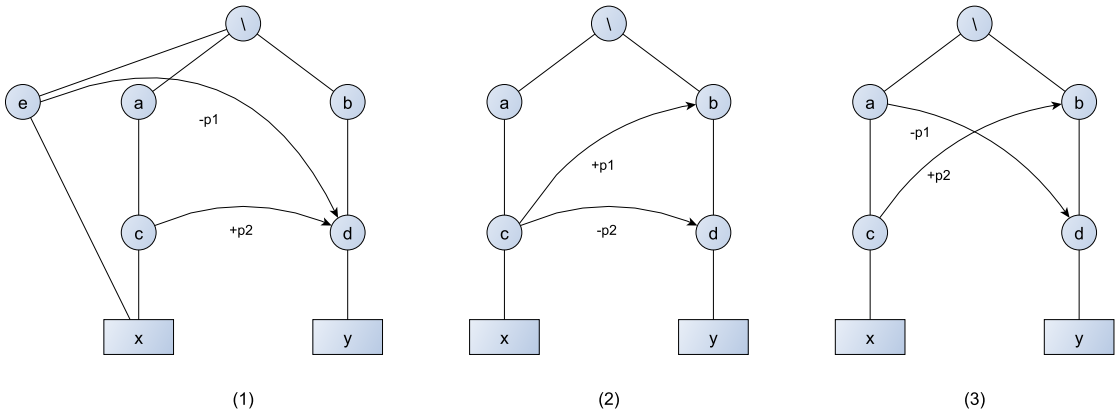


Figure 2.3 – Exemples de conflits

- si p1 et p2 sont normales, alors p2 a précédence sur p1 dans (2) et (3).

Le graphe (1) de la figure 2.3 soulève le problème des règles non comparables par exemple lorsque le sujet est contenu dans plusieurs domaines. Le traitement fait dans

ce cas-là par Ponder diffère de l'algorithme 1. Si les règles ne sont pas comparables, les règles finales intervenant dans le conflit sont traitées comme normales : les règles les plus basses auront précédence.

Donc dans le cas du graphe (1) de la figure 2.3, que p2 ou p1 soient finales ou non, l'interdiction p1 (règle négative) aura toujours précédence sur p2.

2.5 Ponder : test de l'implémentation

Ponder tel que présenté dans les articles [5, 18, 19] et dans les spécifications [4], n'est plus supporté et, à la place, a été mis sur pied Ponder2 [1] dont le principe est globalement le même. Un premier modèle avait été fait en Ponder mais n'avait pu être testé avec le toolkit à cause de son retrait.

Le framework autour de Ponder2 est basé sur 3 composants majeurs :

1. les classes en Java des objets présents dans la hiérarchie de domaine ;
2. le code PonderTalk ;
3. la cellule SMC («*Self Managed Cellule*»).

Des tests ont été effectué sur le framework à partir d'un cas d'étude : le transfert d'un patient d'un hôpital vers un autre.

2.5.1 Les classes en Java

Principe

Pour pouvoir mettre des objets dans la hiérarchie de domaine, il faut d'abord les coder en Java. Chaque objet présent dans la hiérarchie devra être un objet héritant de ManagedObject et devra avoir une référence d'un objet P2Object qui lui sera attribué, car la SMC qui gère la hiérarchie de domaine n'utilise que les références des objets P2Object.

1. http://www.ponder2.net/

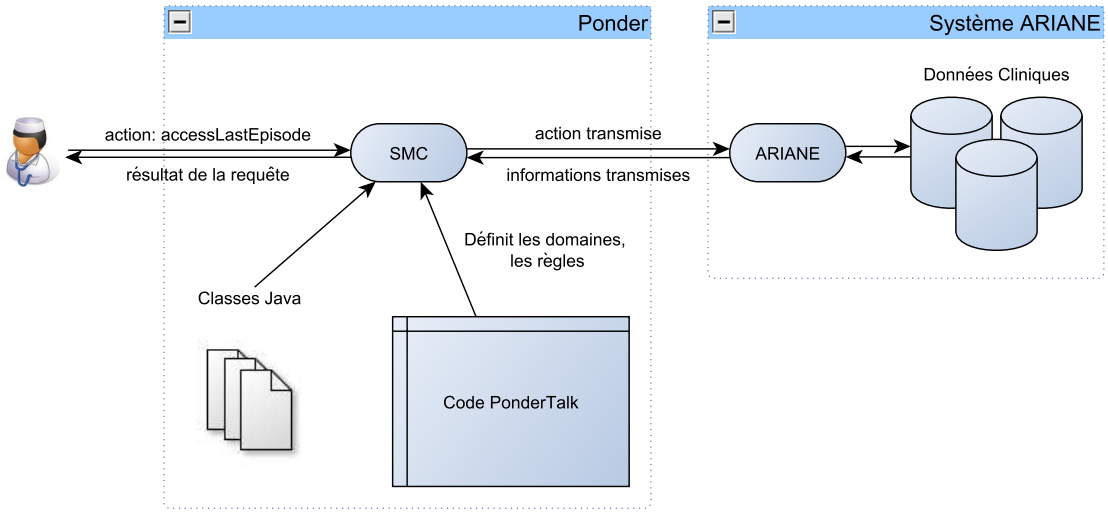


Figure 2.4 – Fonctionnement de Ponder couplé à ARIANE

Test

Le code des classes utilisées pour le test a été mis en annexe. Pour le test, ont été créées :

- une classe Carer, pour les objets représentant les intervenants ;
- une classe Patient, pour les objets représentant les patients ;
- les classes Metarecord, Record et Episode, pour les objets représentant le DMP.

Un patient possède un DMP (ici représenté par le Metarecord) dans lequel sont stockés les dossiers (représentés par Record) sur son sujet de différents établissements. Chaque dossier est constitué d'épisodes.

Par souci de simplicité, les objets issus des classes Metarecord, Record et Episode ne seront pas mis dans la hiérarchie de domaine et seront accessibles via la classe Patient. Les objets instanciés présents dans la hiérarchie de domaine seront donc issus des classes Carer et Patient.

Les principales méthodes de la classe Carer sont :

- getName() : retourne le nom de l'intervenant ;
- accessLastEpisode(P2Object patient,String org) : permet d'accéder au dernier épisode d'un patient dans une organisation donnée ;

- accessPastEpisode(P2Object patient,String org,int index) : permet d'accéder à l'épisode indexé par index dans une organisation donnée ;
- getOrganization() : retourne l'organisation où travaille l'intervenant.

Les intervenants ne sont pas les seuls à pouvoir accéder au dossier d'un patient : le patient doit pouvoir consulter son propre dossier. Les principales méthodes de la classe Patient sont :

- getName() : retourne le nom du patient ;
- accessLastEpisode(String org) : permet d'accéder au dernier épisode du patient dans une organisation donnée ;
- accessPastEpisode(String org,int index) : permet d'accéder à l'épisode indexé par index dans une organisation donnée ;
- getOrganization() : retourne l'organisation où séjourne le patient ;
- setCarer(P2Object Carer) : permet d'affecter un médecin traitant au patient ;
- hasCarer(P2Object Carer) : permet de vérifier si un médecin est le médecin traitant du patient.

Quant aux données, la classe Metarecord est une table de hachage qui fait le lien entre des Record contenant des Episode avec des chaînes de caractères (le nom de l'établissement associé au dossier) ; pour faire simple, Record est une liste contenant des Episode.

2.5.2 Le code PonderTalk

Le code PonderTalk est le code qui permet de spécifier les règles en Ponder. Le langage PonderTalk est basé sur le langage smalltalk [13]. Tous deux sont fortement orientés objets : tout est objet, des chaînes de caractères, aux entiers, les booléens, ainsi que les définitions de classes ; même les blocs de code sont eux aussi représentés en tant qu'objets. En PonderTalk, lors de la manipulation des objets, il est possible de faire appel aux méthodes Java que l'on a implémenté.

Type de message et correspondance

Les types de messages acceptés en PonderTalk sont :

- les messages unaires : ils ne possèdent pas d'argument ;

Tableau 2.1 – Correspondance PonderTalk-Java

PonderTalk	Java
patient getName.	patient.getName()
patient getLastEpisode : CHUS.	patient.getLastEpisode(CHUS)
patient getPastEpisode : 3 org : CHUS.	patient.getPastEpisode(3,CHUS)
carer == patient getCarer.	carer.equals(patient.getCarer)

- les messages binaires : ils possèdent un unique argument, constitué d'un symbole et d'un argument, tel que «== argument» ;
- les messages avec un mot-clef : ils possèdent autant d'arguments que de mots-clefs, sachant que la méthode suivie de «:» peut être considérée comme un mot-clef.

Le tableau 2.1 montre des exemples des correspondances de commandes PonderTalk avec des méthodes Java.

La correspondance se fait techniquement dans le code Java : avant chaque méthode liée à PonderTalk, une annotation est placée et est interprétée lors de la compilation. L'annotation est de la forme :

@Ponder2op("nom_de_la_méthode :arg2 :arg3 :")

Ainsi, dans le code Java nous obtenons :

```
@Ponder2op("accessLastEpisode:org:")
        public P2Object accessLastEpisode(P2Object patient,String organization)
        throws Ponder2Exception{
        ...
        }
```

et lors de l'invocation de la méthode on a :

/root/CHUS/personnel/John accessLastEpisode : /root/CHUS/urgence/lits/Paul org : CHUS.

Ce qu'on peut décomposer comme suit :

/root/CHUS/personnel/John (objet de classe Carer) — accessLastEpisode : (méthode, capture du premier argument)

/root/CHUS/urgence/lits/Paul (objet de classe Patient, premier argument capté) — org : (mot-clef pour le 2nd argument) — CHUS (2nd argument) .

```
//affectation d'un bloc a la variable message
message := [ root print: "Hello, World!"].
//evaluation du bloc
message value.

-> Hello, World !
```

Figure 2.5 – Exemple d'utilisation d'un bloc

Objets basiques

Même un domaine est un objet, et correspond en fait à une table de hachage : domain at : «*clef*» put : *objet*. On insère donc un objet dans un domaine en se servant des mots clefs at : put :. Pour récupérer un objet d'un domaine il suffit d'utiliser at : «*clef*».

Les blocs («*blocks*») permettent d'encapsuler plusieurs déclarations dans un objet. Les blocs peuvent avoir des arguments optionnels qui permettent d'utiliser le bloc comme une procédure stockée. Un bloc est de la forme [:arg1 :arg2 | declaration1. declaration2]. Un bloc étant un objet, il est possible de l'évaluer et celui-ci retourne ce que retourne la dernière déclaration. La figure 2.5 fournit un exemple d'utilisation.

Il est possible d'utiliser des tableaux : la figure 2.6 illustre le traitement d'un tableau services contenant des chaînes de caractères. Un tableau a pour syntaxe #(ele1 ele2 ele3). Le mot-clef do : suivi d'un bloc permet d'exécuter le bloc pour chaque élément du tableau passé en paramètre au bloc.

Ainsi dans la figure 2.6, pour chaque élément du tableau capturé par la variable name, il est créé un domaine qui contiendra les domaines bed et personnel.

Enfin les règles sont également des objets. Ce sont des objets de type authpolicy.

Pour définir une règle complètement, il est nécessaire de savoir si la règle est valable pour le sujet ou bien pour la cible ou encore pour les deux. Le mot-clef focus : permet de définir la portée de la règle.

– focus : "t" définit la règle valable pour la cible. Le point important est donc la réponse.

```
services do: [ :name |
        ward := hospital/ward at: name put: domain create.
        ward at: "bed" put: domain create.
        ward at: "personnel" put: domain create.
].
```

Figure 2.6 – Exemple d'utilisation de tableaux en PonderTalk

- focus : "s" définit la règle valable pour le sujet. Le point important est donc la requête.

Bien sûr, il est possible de définir :

- un sujet (subject :) ;
- une cible (target :) ;
- une action (action :) ;
- un état d'activation pour rendre la règle active (par défaut elle est inactive, donc non prise en compte par les PEP).
- une modalité pour la requête ou une modalité pour la réponse, suivant le choix du focus(repneg. ou reqneg.) ;
- une clause de condition (repcondition : ou reqcondition : selon le PEP).

Test

Dans notre cas d'étude, le code Pondertalk est principalement constitué :

- du code pour la création de fonctions ;
- des règles.

La figure 2.7 contient une fonction d'ajout d'hôpital à la hiérarchie de domaine, prenant en argument le nom de l'hôpital (:name) et une liste de services (:services).

La figure 2.8 montre un exemple de création d'une règle que l'on place dans le domaine root/policy et qui interdit à n'importe quel objet d'effectuer l'action data sur n'importe quel objet.

La règle présentée à la figure 2.9 supplante la règle négative de la figure 2.8 si la cible a bien pour intervenant-traitant le sujet.

```
//creation d'une fonction d'ajout d'hopital
addhospital := [ :name :services |
    hospital := root/hospital.
        hospital := root/hospital at: name put: domain create.

        // Creations des services et du personnel
        hospital at: "ward" put: domain create.
        hospital at: "personnel" put: domain create.

        //Parcours du tableau services
        //Creation des services
        services do: [ :name |
                ward := hospital/ward at: name put: domain create.
                ward at: "bed" put: domain create.
                ward at: "personnel" put: domain create.
        ].
].

//ajout de la fonction dans la hierarchie
//dans le domaine root/sc
root/sc at: "addHospital" put: addhospital.
```

Figure 2.7 – Fonction de création de domaine d'hôpital en PonderTalk

```
newauthpol := root/factory at: "authpolicy".

root/policy at: "global_deny_data" put:
                        (newauthpol subject: root
                                action: "data"
                                target: root
                                focus: "s").
root/policy/global_deny_data reqneg.
root/policy/global_deny_data active: true.
```

Figure 2.8 – Exemple d'interdiction en PonderTalk

```
root/policy at: "rule_traiting_patient" put:
                        (newauthpol subject: root
                                    action: "data"
                                    target: root
                                    focus: "s").
root/policy/rule_traiting_patient
        reqcondition: [p_target HasCarer: p_subject].
root/policy/rule_traiting_patient final.
root/policy/rule_traiting_patient active: true.
```

Figure 2.9 – Exemple d'autorisation en PonderTalk

2.5.3 La cellule SMC

La cellule SMC contient les PEP et le PDP. Le dialogue avec la cellule se fait via Telnet avec le langage PonderTalk. Lors du lancement de la cellule, il faut préciser la modalité par défaut. Il est possible de charger des fichiers contenant du code PonderTalk pour, par exemple, initialiser la hiérarchie de domaine. Il est utile de placer les différentes fonctions chargées dans la hiérarchie de domaine, car les variables sont propres à chaque fichier et ne sont pas valides en dehors. Ainsi la fonction addHospital qui se trouve dans le fichier fonctions.p2 ne peut être appelée que si elle est dans la hiérarchie de domaine !

Lors d'une interrogation de la SMC, par exemple en faisant une action telle que patient data., la SMC ne consulte pas à chaque fois les règles. Le contrôle ne s'applique que par l'appel de la méthode operation existante dans la classe P2Object. Cette méthode, qui peut soit renvoyer le résultat (cas où l'action est autorisée) soit lever une exception, s'utilise de la façon suivante :

```
cible.operation(sujet,"action","liste_params")
```

Le fait que operation prenne comme cible un P2Object oblige l'existence d'une référence P2Object reliée à la cible. Lors de l'appel d'un constructeur en Java via l'invocation PonderTalk, il est possible d'attribuer une référence P2Object à un objet tel que le montre le programme 2.1. L'attribution de la référence P2Object se fait lors de l'invocation PonderTalk : pour cela, il faut obligatoirement que le premier

argument du constructeur soit nommé «myP2Object».

programme 2.1 – Constructeur de la classe Carer

```
@Ponder2op("create:")
public Carer(P2Object myP2Object, String name){
        this.myP2Object = myP2Object;
        this.name=name;
}
```

Ainsi la commande patient data. fonctionnera toujours alors qu'une invocation de la méthode access présentée par le programme 2.2 par la commande «intervenant access : patient.» risque d'échouer car elle déclenchera l'évaluation des règles.

programme 2.2 – Méthode access

```
@Ponder2op("access:")
public P2Object access(P2Object patient) throws
        Ponder2ArgumentException,
        Ponder2OperationException,
        Ponder2Exception{
        P2Object data = patient.operation(myP2Object,"data");
        return data;
        }
```

2.6 Test et Conclusions

La phase de test s'est déroulé sur le cas d'un patient transféré d'un hôpital à un autre. Cela a permis de montrer certains handicaps.

Surcharge de la hiérarchie de domaine

En plus des cibles et sujets, les règles elles-mêmes doivent être obligatoirement dans la hiérarchie de domaine. Il en va de même pour toutes les fonctions de gestion de la hiérarchie et des constructeurs d'objet. Cela constitue un handicap majeur pour

notre projet car chaque donnée du patient devra être mise dans la hiérarchie qui donc deviendrait un reflet redondant des bases de données existantes.

Manque d'expressivité

PonderTalk manque d'expressivité. En effet, pour la spécification des clauses des règles, il n'est possible d'utiliser que :

- les arguments passés à l'action ;
- le sujet via la variable globale P2_subject ;
- la cible via la variable globale P2_target.

Les deux variables globales P2_subject et P2_target ont été ajoutées à Ponder seulement après contact avec les auteurs car la spécification des clauses était particulièrement ardue. Ainsi il n'est pas possible d'accéder simplement à des variables de l'environnement.

Un autre point important est la limite qu'introduit le langage PonderTalk pour la création des différentes fonctions nécessaires à la gestion de la hiérarchie : par exemple déplacer un patient n'est pas toujours chose aisée.

Trop orienté «implémentation»

L'intégration PonderTalk/Java ne correspond pas à nos besoins : il nous faudrait concevoir une couche supplémentaire par dessus l'application existante, sans avoir accès au code complet de l'application ARIANE.

Filtrage

La notion de filtrage est inexistante au niveau de la spécification des règles. Il n'est pas possible de filtrer une requête, le résultat renvoyé par le PDP étant binaire : autorisation ou interdiction.

Conclusion

Bien que Ponder possède plusieurs aspects négatifs qui rendent la solution inappropriée, des concepts intéressants sont réutilisables pour notre solution :

– la hiérarchie de domaine ne peut être utilisée telle quelle mais apporte la notion d'héritage ;
– bien qu'incomplète, la stratégie de résolution de conflit semble prometteuse.

Chapitre 3

Modélisation du consentement au CHUS

Les solutions étudiées ne satisfaisant pas nos besoins, la modélisation des règles est nécessaire afin de pouvoir monter un prototype et surtout comprendre leur fonctionnement. Les concepts importants adjacents au DMP sont regroupés au sein d'un graphe de concept. Une analyse de l'architecture envisagée est faite. La traduction des règles en langage naturel plus formel permet de dégager des patrons de règles qui serviront au prototype.

3.1 Graphe de concepts

Le graphe de concepts associé à notre problème est représenté sur la figure 3.1.

Patient

On distingue plusieurs types de patients : suivant la gravité de leur maladie, le consentement ne sera pas géré de la même façon. Par exemple, le consentement d'un patient atteint de la maladie d'Alzeimer n'est pas géré de la même manière que le consentement d'un patient souffrant d'une allergie. On compte également le patient réfséré : le patient est envoyé par un médecin prescripteur afin d'effectuer des examens complémentaires que le médecin ne peut effectuer lui-même. Dans ce cas-là, les données issues des examens sont envoyés au mé-

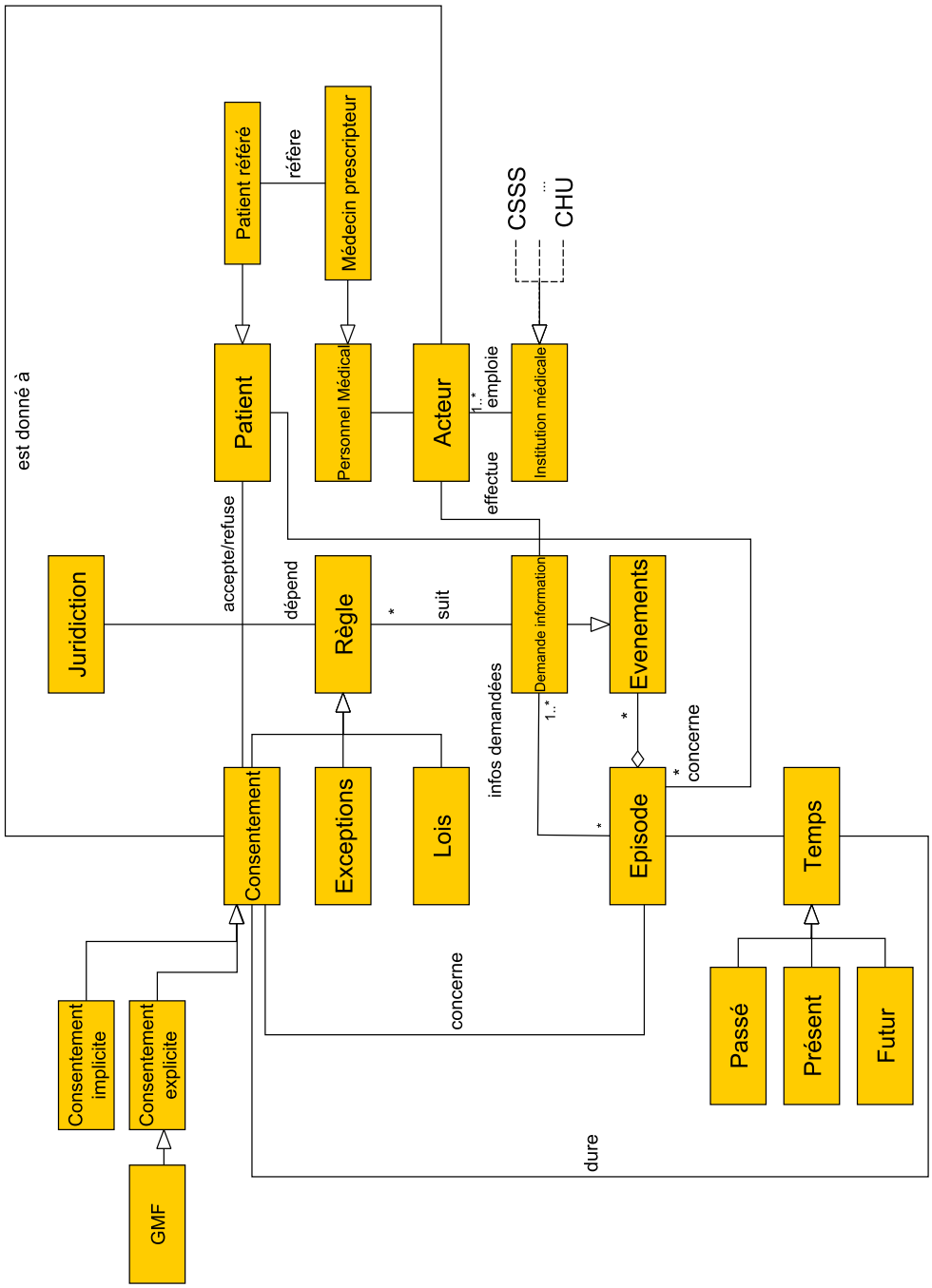


Figure 3.1 – Graphe de concepts

decin prescripteur, les données lui appartenant ; le consentement du patient est implicite puisqu'il se présente à l'hôpital pour les examens.

Épisode

Un épisode contient toutes les données à partir du moment où le patient se présente à l'hôpital jusqu'au moment où il le quitte. Un épisode peut également être appelé «visite».

Consentement

Le consentement porte sur :

- les données que l'on a choisi de partager ;
- les personnes ou les entités à qui l'on permet de prendre connaissance du dossier médical ;
- la période sur laquelle on permet aux différents acteurs de consulter le dossier médical.

On peut le décomposer en deux :

- consentement implicite : c'est le consentement qui est donné lors de l'accès aux soins ; au début de l'épisode, le patient doit signer une feuille qui indique qu'il consent à recevoir des soins mais également à communiquer ses informations médicales relatives à ces soins.
- consentement explicite : c'est le consentement que le sujet doit recevoir du patient pour avoir accès à son dossier en dehors de l'épisode ou lorsque le sujet n'est pas le médecin traitant du patient. Par exemple, le consentement GMF (Groupe des Médecins de Famille), qui est renouvelable tous les ans auprès du médecin de famille GMF permet à n'importe quel médecin GMF d'avoir accès aux données du patient : c'est un consentement explicite.

Temps

Le temps est décomposé en 3 : passé, présent, et futur car il est possible de définir son consentement pour une période indéterminée (présent, futur) ou déterminée (un épisode passé).

Événement

Un événement est ce qui compose un épisode. En effet, un épisode est constitué de toutes les données concernant le patient à partir du moment où il vient à l'hôpital jusqu'au moment où il le quitte. Ainsi, un patient qui viendrait une fois

pour une radiographie et une opération et qui reviendrait pour de la rééducation aura connu 2 épisodes même si le premier épisode est constitué de 2 événements.

Exceptions

Les exceptions sont les cas définis par la loi, et permettent d'avoir accès au dossier médical du patient sans avoir besoin de son consentement. Ce sont les mêmes définis à la section 1.1.2.

Lois

Les lois spécifient les cas de nécessité du consentement, en définissant les exceptions. Elles permettent également de gérer au niveau légal la protection des informations en associant aux devoirs des sanctions.

Juridiction

Notre solution étant voulue la plus générale possible, doit être adaptable dans des provinces où les législations diffèrent.

Institution médicales

Les institutions médicales au Québec sont découpées comme le montre la figure 3.2 :

- Ministère de la Santé et des Services Sociaux ;
- Agence de la Santé et des Services Sociaux ;
- Réseau Local de la Santé et des Services Sociaux : l'Agence de la Santé et des Services Sociaux se décompose en plusieurs RLS ;
- Tout le reste, qui dépend du RLS associé : c'est ce que l'on aimerait interfacer dans un premier temps, afin que les différents établissements dépendants du RLS partagent leurs informations.

3.2 Architecture possible

La gestion du contrôle des accès et la gestion du consentement du patient nous pousse à adopter une architecture où :

- les différentes politiques d'accès devront être stockées dans une base de données ;
- une entité pourrait pouvoir modifier/créer ces politiques d'accès ;
- une entité déciderait d'autoriser ou d'interdire une requête selon les politiques ;

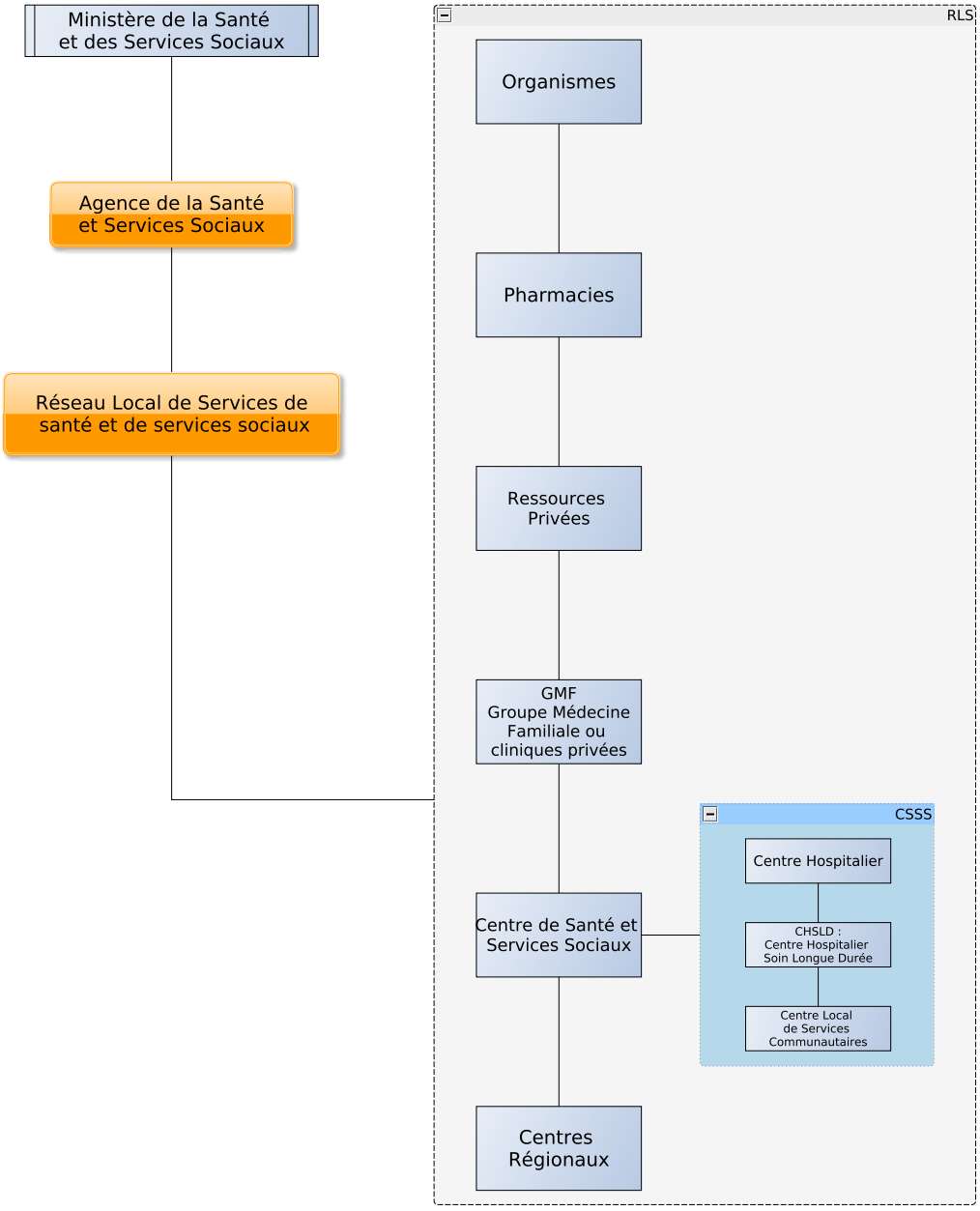


Figure 3.2 – Système de la santé au Québec

– une entité veillerait à ce que les politiques décidées soient appliquées et gérerait les conflits.

L'architecture des politiques d'accès IETF, sur laquelle est basé Ponder, conviendrait à nos critères comme le montre la figure 3.3 et est proche de l'architecture de XACML (figure 1.8).

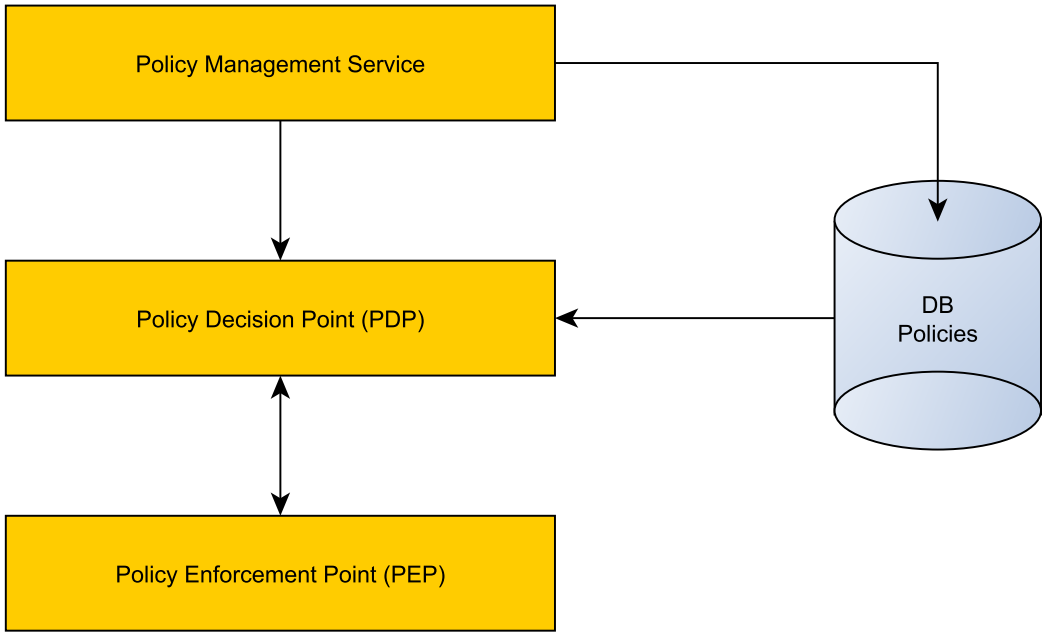


Figure 3.3 – Architecture des politiques d'accès IETF

Le PDP (Policy Decision Point) choisit les politiques d'accès et les règles à appliquer et comment elles le seront selon les autres politiques en vigueur et les autres données.

Le PEP (Policy Enforcement Point) applique les politiques. Il doit :

- vérifier l'exécution des applications et le déclenchement des évaluations des politiques aux moments appropriés ;
- collecter les informations nécessaires à l'évaluation des politiques ;
- appliquer les décisions du PDP.

Le PMS (Policy Management Service) ou encore **PAP** (Policy Administration Point) est une interface pour administrer, spécifier, éditer les politiques d'accès.

Cette architecture pourrait s'appliquer à bien des situations de contrôle d'accès. Dans le cas de la gestion du consentement, le schéma pourrait ressembler au diagramme de la figure 3.4.

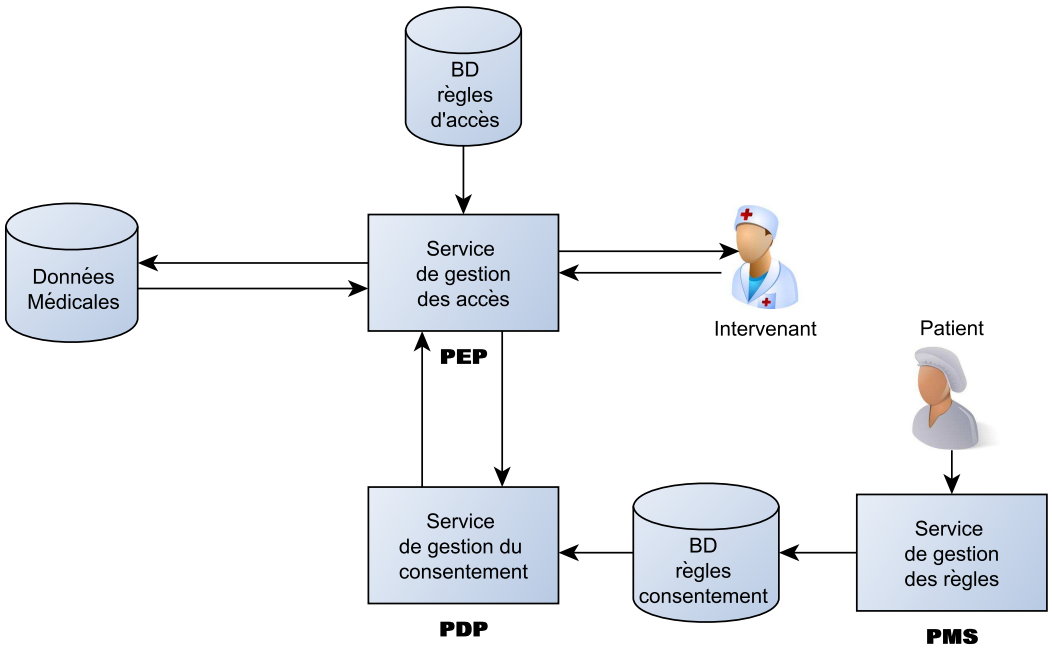


Figure 3.4 – Exemple d'architecture appliquée à la gestion du consentement

Sur cette figure, le PMS permet au patient de gérer son consentement, permettant à celui ci d'insérer, de modifier, supprimer les règles dans la base de données. Le service de gestion d'accès intercepte la requête du requérant, et transmet les informations de la requête vers le service de gestion du consentement qui a le rôle du PDP. Celui-ci prend en compte la base de règle mise en place par le patient via le PMS et retourne la décision au service de gestion d'accès qui l'applique. Les règles d'accès et le consentement ne se trouvent pas sur la même base de données car chaque demande d'information doit être justifiée par la nécessité. Il est donc normal d'instaurer un filtre par profil d'accès : une infirmière n'aura pas les mêmes droits qu'un médecin et n'aura pas accès à certaines informations. Le consentement est complémentaire

des règles d'accès. Il provient de l'usager ou bien de son représentant légal dans le cas où l'usager ne pourrait être en état de donner son consentement, lequel permet à une personne accréditée (qui possède donc un profil adéquat) d'avoir accès aux informations.

3.3 Règles en langage naturel

Dans un premier temps, les règles décrites dans la sous-section1.3.1 sont retranscrites en langage naturel afin d'être plus facilement formalisées. Quelque soit la règle, l'accès est restreint aux données visibles selon les profils d'accès, mécanisme déjà en place au sein du CHUS.

Transfert de service

Soient A,B des établissements différents, a un intervenant de l'établissement A et b un intervenant de l'établissement B, x un patient.

1. x est admis dans l'établissement A.
2. Si a fait partie des intervenants traitant x, alors a a accès à l'épisode en cours de x.
3. a demande un transfert de x vers B.
4. x est transféré dans l'établissement B.
5. L'épisode de x dans A se ferme : a n'a plus accès à l'épisode en cours de x.
6. Un épisode dans l'établissement B s'ouvre pour x.
7. Si b fait partie des intervenants traitant x, alors b a accès à l'épisode en cours de x et b a accès à l'épisode passé de x dans A.
8. Quand x quitte B, l'accès pour b à l'épisode passé de x dans A se termine et l'accès pour b à l'épisode de x dans B se termine.
9. L'épisode de x dans B se ferme.

Transfert vers un CRE

Soient A un établissement, B un CRE, a un intervenant de l'établissement A et b un intervenant de l'établissement B, x un patient et c le personnel en charge d'étude des admissions.

1. x est admis dans l'établissement A.
2. Si a fait partie des intervenants traitant x, alors a a accès à l'épisode en cours de x.
3. Une demande de transfert est demandée pour x de A vers B.
4. A demande un consentement à x pour l'accès aux informations concernant l'épisode passé e dans A.
5. Si x accepte, c a accès à e.
6. Si B ne peut pas gérer x, c n'a plus accès à e.
7. Sinon
   - x est transféré dans l'établissement B.
   - L'épisode de x dans A se ferme.
   - a n'a plus d'accès à l'épisode en cours de x.
   - c n'a plus accès à e.
   - Un épisode dans l'établissement B s'ouvre pour x.
   - Si b fait partie des intervenants traitant x, alors b a accès à l'épisode en cours de x et b a accès à e.
   - Quand x quitte B, l'accès pour b à e se termine, l'accès pour b à l'épisode de x dans B se termine. L'épisode de x dans B se ferme.

Médecin appartenant à un GMF

Soit x un patient, m un médecin appartenant à un GMF auquel a souscrit x. Quelque soit e épisode de x, m a accès à e.

Envoi des AH-280

Soit x un patient admis en urgence ou clinique externe pour un épisode e dans un établissement E. Soit t le médecin traitant de x dans E, f le médecin de famille de x.

Tant que x est dans E pour l'épisode e, t a accès à l'épisode e.

Si x désire que f ait accès aux données contenues dans l'AH-280 alors f peut accéder à ces données.

Patient décédé

Soit x un patient.

Si x décède, alors quelque soit i intervenant non archiviste, non chercheur, non DSP, i n'a pas accès aux données de x.

Clinique du 24 juin

Soit m un médecin de la Clinique du 24 Juin et x un patient de la Clinique 24 Juin.

Si x autorise l'accès, alors quelque soit e épisode de x dans le CHUS,

si m est un médecin traitant de x, alors m a accès à e.

IUGS

Soit m un médecin de l'IUGS du 24 Juin et x un patient de l'IUGS.

Si x autorise l'accès, alors quelque soit e épisode de x dans le CHUS,

si m est un médecin traitant de x, alors m a accès à e.

Hémodialyse

Soit m un médecin du CHUS et x un patient du CSSS de Magog transféré du CHUS.

Si x autorise l'accès via le formulaire de consentement Hémodialyse, alors quelque soit e épisode de x dans le CHUS,

si m est un médecin traitant de x alors m a accès à e.

Délégation

Soient x un patient, y une personne, z un intervenant.
Si x délègue son consentement à y pour un ensemble d'évènements e,
si y autorise l'accès de z à e, alors z a accès à e.

Transfert inter-établissement

Soient A, B des établissements différents, b un intervenant de l'établissement B, x un patient de B.
Si x autorise l'accès pour un ensemble d'évènements e dans A,
si b est un médecin traitant de x, alors b a accès à e.

3.4 Patrons de règle

Afin de comprendre comment interagissent les règles entre elles, il est utile de créer des patrons de règles. Ces patrons sont faits à partir des cas étudiés, de la littérature et également à partir des entrevues avec les archivistes travaillant au CHUS. On peut répertorier les règles issues de la loi, appelées *exceptions*, les règles issues du libre arbitre du patient, appelée *règles explicites* ou encore les règles issues de la pratique dans le domaine hospitalier, appelées *règles implicites*.

3.4.1 Règles explicites

On appelle règles explicites les règles que le patient a formulé *explicitement*.

Autorisation spécifique
Le patient autorise une personne, une entité qui ne devrait pas avoir accès à son dossier à y avoir accès. Cette autorisation contourne le principe de nécessité qui restreint justement l'accès aux données. Ce type de règles peut être rencontré dans des cas où un patient aimerait connaître l'avis d'un tiers par exemple.

Interdiction sur une donnée spécifique
Le patient choisit d'interdire l'accès à certaines données auxquelles certaines entités devraient y avoir droit. Par exemple, un patient pourrait choisir de masquer ses épisodes médicaux en relation avec ses données psychiatriques.

Interdiction sur une entité spécifique
Le patient choisit d'interdire l'accès à des entités spécifiques, même si celles ci devraient être autorisées. Par exemple, un patient peut interdire à une de ses connaissance l'accès à son dossier.

Restrictions
Outre les interdictions et les autorisations, le patient peut spécifier des restrictions. Les restrictions se distinguent des interdictions et des autorisations par le fait qu'une restriction est à la fois une interdiction et une autorisation. Par exemple, un patient peut n'autoriser *que* les infirmières-chefs (*ensemble restreint*) *parmi* les infirmières (*ensemble à restreindre*) à avoir accès à son dernier épisode. Ici la restriction se fait sur les sujets et peut se décomposer comme :

- une autorisation pour les infirmières-chefs ;
- une interdiction pour toutes les infirmières qui ne sont pas infirmières-chefs.

Ces restrictions peuvent porter soit sur les sujets, soit sur les données. Dans le cas d'une restriction sur les sujets, le patient n'autorise ou interdit qu'une partie du personnel médical à avoir accès à son dossier. Il est plus courant de restreindre l'accès à une partie du dossier à un sous-ensemble du personnel médical que d'interdire chaque intervenant n'appartenant pas à ce sous-ensemble.

Dans le cas d'une restriction sur les données, le patient n'autorise qu'une partie de ses données à être consultable, les données n'appartenant pas à cet ensemble n'étant pas accessibles.

Il faut noter qu'une restriction porte sur un *ensemble à restreindre* et *un ensemble restreint*. Une restriction peut être une autorisation restreinte (autorisation restreinte à un sous-ensemble) ou une interdiction restreinte (interdiction restreinte à un sous-ensemble).

3.4.2 Règles implicites

On appelle règles implicites les règles auxquelles le patient consent lors d'une visite de soin. Ces règles ne sont pas explicitées au patient, mais celui-ci consent à *toutes* ces règles dans le cadre des soins reçus. Cette sous-section présente quelques exemples de règles en vigueur dans le domaine hospitalier.

Restriction au médecin traitant lors d'une visite
À chaque épisode est associé un médecin traitant. Un médecin est qualifié de médecin traitant durant la visite seulement. Seul ce médecin traitant a accès à l'épisode associé, lorsqu'il est ouvert.

Autorisation lors d'un transfert
Lors d'un transfert, l'équipe d'intervenants du service a accès au dernier épisode de celui-ci dans le service d'origine et à l'épisode en cours (dans le service d'arrivée). Cette autorisation est généralement couplée avec une restriction au niveau du médecin traitant.

Délégation
Le médecin traitant peut autoriser des personnes tierces à avoir accès à une partie du dossier auquel il est associé, afin de déléguer une tâche, demander conseil. L'autorisation est valide tant que le médecin traitant est autorisé à avoir accès au dossier, donc la plupart du temps pour la durée de la visite.

3.4.3 Synthèse

La «classification» des règles comme étant des règles implicites, explicites ou des exceptions (les exceptions, présentées dans la section 1.1.2, sont en intégralité des autorisations qui contournent toutes les autres règles) permet d'esquisser un modèle d'«ordonnancement» des règles. En effet on voit que les exceptions prévalent sur n'importe quelle autre règle et que les règles explicites sont plus prioritaires que les règles implicites. Une règle explicitement donnée par le patient doit prendre le pas sur

une règle implicite qui serait comparable à une règle d'accès. Si on devait modéliser une règle par un objet, une règle aurait pour attribut :

- un sujet ;
- une cible ;
- une action ;
- une modalité ;
- une clause ;
- un ensemble restreint (à spécifier seulement pour une restriction), qui peut être une cible ou un sujet ;
- un niveau de priorité.

Chapitre 4

Algorithme de gestion de conflit

Le chapitre précédent a permis de mettre en évidence une certaine précédence entre les divers types de règles. Cela ne permet pas toutefois de résoudre les conflits au sein d'un même type de règle : un algorithme de gestion de conflit est requis. S'appuyant sur le mécanisme de gestion de conflit vu en Ponder, l'algorithme proposé est plus général que celui de Ponder, ne se limitant pas à deux niveaux de priorité (les règles normales et finales) mais de plus permet d'incorporer la notion de filtrage qui fait défaut en Ponder. L'algorithme s'appuie sur quelques concepts de Cassandra et sur une hiérarchie de profils et une hiérarchie de données que nous expliquerons par la suite.

4.1 Cassandra

Cassandra est un langage de spécification basé sur Datalog, sous-ensemble de Prolog. C'est un langage fortement déclaratif développé par Cassassa Mont *et al* [15]. L'étude approfondie de ce langage a été effectué par Thierry Le Ba dans le cadre de sa maîtrise et du projet avec le CHUS. Le fonctionnement complet est détaillé dans son mémoire. Pour la suite, il sera utile de savoir que Cassandra est inspiré de RBAC [9] et possède comme RBAC des rôles. Chaque entité devra être connectée au système avec un rôle actif. Ainsi un médecin, lorsqu'il devient médecin traitant d'un patient, endosse le rôle actif de médecin traitant et le quittera lorsqu'il ne sera plus médecin traitant. Chaque entité ne peut avoir qu'un rôle actif à la fois.

4.2 Hiérarchie de profils et hiérarchie de données

Fort des expérimentations en Ponder, l'idée des hiérarchies a été reprise. L'idée est de constituer deux hiérarchies séparées afin de ne pas être surchargé comme en Ponder. On aurait alors une hiérarchie de profils qui ne contiendrait que des profils et une hiérarchie de données qui contiendrait les données. Cela reviendrait à scinder la hiérarchie de domaine de Ponder en deux, séparant les sujets des cibles.

On pourrait créer une hiérarchie de profil avec le principe suivant : un profil fils hérite des permissions du profil père. Ainsi un profil cumule les permissions de ses ascendants. Par exemple, sur la figure 4.1, on voit que l'infirmière en chef possède toutes les permissions accordées aux infirmières et aux assistantes infirmières. Il en est de même pour les infirmières en urgence. Le fait que les infirmières en urgence et les infirmières chef sont sur le même niveau ne signifie pas qu'elles aient les mêmes permissions.

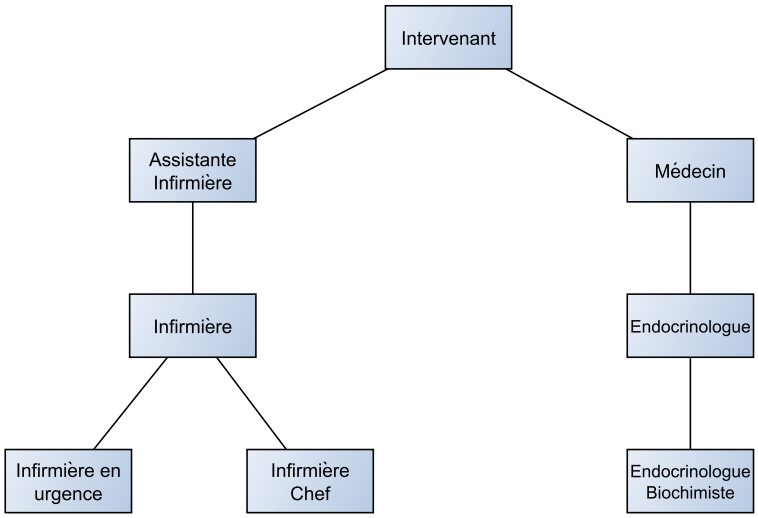


Figure 4.1 – Exemple de hiérarchie de profil

Ce type de hiérarchie est rendu possible par la structure des profils utilisés au sein du CHUS : les profils «spécifiques» se démarquent des profils «généraux» par des «profils secondaires». Ainsi une infirmière en chef diffère d'une infirmière par ses

droits additionnels d'infirmière en chef.

On pourra poser la notation suivante :

$$\textit{infirmière en chef} \text{ descend de } \textit{infirmière} \iff \textit{infirmière en chef} < \textit{infirmière}$$

Par extension on peut introduire $\leq$:

$$x \text{ descend de } y \vee (x = y) \iff x \leq y$$

Concernant la hiérarchie de données, on pourrait s'appuyer sur le principe de découpage proposé par la figure 4.2.

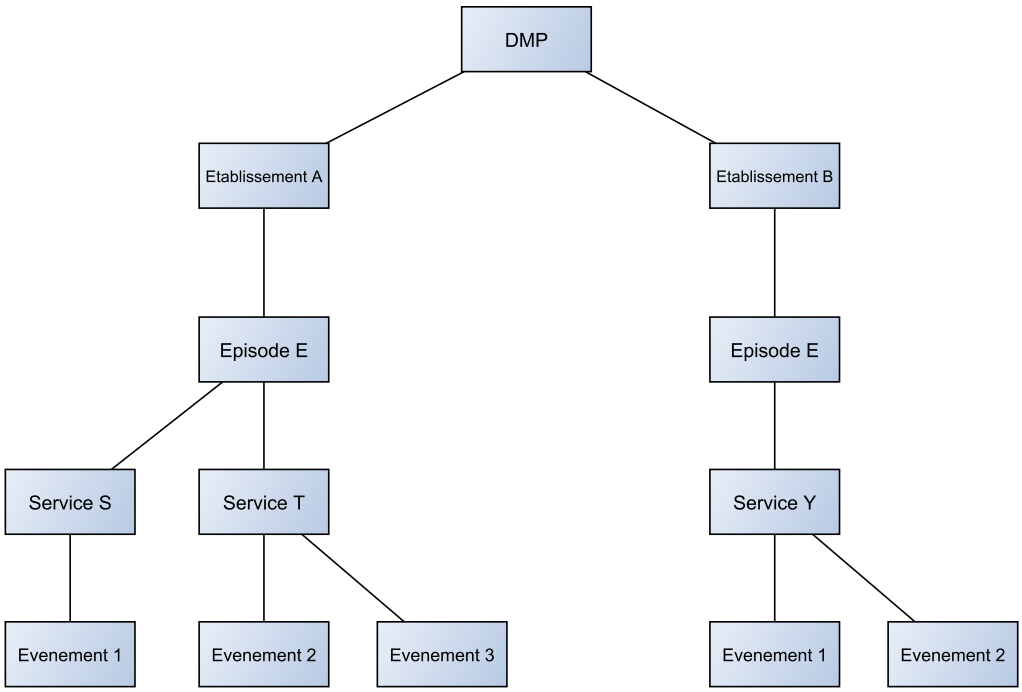


Figure 4.2 – Exemple de hiérarchie de données

On cherche en fait à faire en sorte qu'une donnée fille soit plus «spécifique» qu'une donnée mère. De la même façon on peut réutiliser la notation introduite plus haut :

$$\textit{événement} \text{ descend de } \textit{visite} \iff \textit{événement} < \textit{visite}$$

Ainsi on peut établir une relation de spécificité, notée $<$, parmi les règles : une règle r1 est plus spécifique qu'une règle r2 si et seulement si le sujet de r1 est un descendant du sujet de r2 ou lorsque le sujet de r1 est le même que r2, la cible de r1 est un descendant de la cible de r2. Soit, en réutilisant la notation :

$$r1 < r2 \iff r1.\textit{sujet} < r2.\textit{sujet} \vee (r1.\textit{sujet} = r2.\textit{sujet} \wedge r1.\textit{cible} < r2.\textit{cible})$$

Dans le cas où deux règles ne seraient pas comparables avec $<$, la règle négative a préséance sur l'autre. On pose :

$$r1.\textit{modalite} \leqslant r2.\textit{modalite} \iff r1.\textit{modalite} = \text{interdiction} \vee r1.\textit{modalite} = r2.\textit{modalite}$$

On définit donc $\precsim$ pour comparer deux règles en prenant en compte la spécificité et la modalité :

$$r1 \precsim r2 \iff (r1 < r2) \vee ((r1 \not< r2) \wedge (r2 \not< r1) \wedge r1.\textit{modalite} \leqslant r2.\textit{modalite}) \vee (r1 \cong r2)$$

avec :

$$r1 \cong r2 \iff ((r1.\textit{sujet} = r2.\textit{sujet}) \wedge (r1.\textit{cible} = r2.\textit{cible}))$$

On peut définir également la relation $\sim$:

$$r1 \sim r2 \iff r1 \precsim r2 \wedge r2 \precsim r1$$

Ce qui permet de définir une relation d'ordre $\leq$:

$$r1 \leq r2 \iff (r1 \sim r2) \vee r1 \precsim r2$$

Le problème soulevé par le graphe (1) de la figure 2.3 n'interfère plus car dans Cassandra, lorsqu'un sujet effectue une action il ne peut avoir qu'un seul rôle actif.

Notre hiérarchie de profil peut être donc considérée comme un arbre.

4.3 Gestion de conflit

Ainsi dans la section précédente, a été introduite la notion de «spécificité» qui permet de comparer 2 règles. Cette relation, à première vue, permettrait de donner précédence à une règle plutôt qu'à une autre : par exemple, si on instaure une interdiction globale puis une autorisation locale, on voudrait *a priori* que l'autorisation l'emporte au niveau local. Toutefois, ce n'est pas forcément ce que l'utilisateur veut : par exemple, dans un cas d'urgence, un accès à une partie du dossier doit être permis malgré les différents masquages dans cette partie. Ce cas-ci est beaucoup plus rare que le premier et intervient surtout dans des contextes d'urgence.

Considérant le travail effectué sur les règles, il apparaît que la relation de «spécificité» permettrait de régler la majeure partie des conflits. Mais comme on l'a montré avec le contre-exemple, cela peut échouer, d'où la nécessité d'un algorithme de gestion de conflit.

On pourrait dans un premier temps, classer les règles par niveau de priorité comme l'ont été les patrons de règles dans le chapitre précédent.

1. Les exceptions : les règles qui permettent d'accéder au dossier d'un patient sans son consentement, les lois par exemple.
2. Les règles explicites : les règles expressément spécifiées par le patient.
3. Les règles implicites : les règles issues de la pratique médicale, auquel le patient doit consentir afin de bénéficier des soins.

Dans cet exemple, on voit que les niveaux sont rangés du plus prioritaire au moins prioritaire.

L'algorithme qui est présenté permet de construire, lors d'une demande d'accès à une donnée, le sous-ensemble de la donnée demandée – auquel l'utilisateur a accès, selon les règles définies. On pourra considérer les données comme des ensembles, et on ne traite pas pour l'instant les règles de type restriction.

Pour prendre en charge les restrictions, il suffit de les ré-écrire en deux règles de modalité différente. Prenons l'exemple de la figure 4.2 avec 3 règles classées par

Algorithme 2 Algorithme de gestion de conflit du prototype

Entrées : *regles* : ensemble des règles
requête <$action(sujet, cible)$>
$\leq$: relation d'ordre entre les règles
Sorties : *aut* : ensemble des données autorisées
$regles\_applicables = \emptyset$; aut $= \emptyset$; int $= \emptyset$;
pour chaque élément r de *regles* **faire** {Récupération des règles applicables}
si $sujet \leq r.sujet$ **et** $cible \leq r.cible$ **et** $action \in r.action$ **et** $r.clause$ **alors**
$regles\_applicables$.ajout(r) ;
finsi
fin pour
pour chaque élément r de $regles\_applicables$ **faire** {Classement des règles par niveau}
$niveau[r.niveau]$.ajout(r) ;
fin pour
pour $i = 0 \rightarrow niveauMax$ **faire** {0 est le niveau le plus prioritaire}
$niveau[i]$.tri() {Classement décroissant des règles selon $\leq$ }
$aut_i = \varnothing$, $int_i = \varnothing$
pour chaque règle r de $niveau[i]$ **faire**
si $r.modalite ==$ autorisation **alors**
$aut_i = aut_i \cup (r.cible \setminus int_i)$
sinon
$int_i = int_i \cup r.cible$
finsi
fin pour
$aut = aut \cup (aut_i \setminus int)$
$int = int \cup int_i$
fin pour
$aut = aut \cap cible$ {Restriction aux données demandées}

spécificité qui ont le même niveau de priorité :

1. interdiction sur les infirmières d'accéder à l'événement 2 du service T de l'épisode E de l'établissement A, appelé DAET2 par la suite ;
2. restriction concernant les infirmières : elles ne peuvent accéder qu'aux données issues du service T de l'épisode E de l'établissement A, appelées DAET par la suite, parmi les données de l'établissement A, appelées DA par la suite ;
3. autorisation sur les infirmières d'accéder aux données du service S de l'épisode

E de l'établissement A, appelées DAES par la suite.

Comme vu précédemment en 3.4.1, une restriction peut se décomposer en deux règles de modalités différentes : il suffit de ré-écrire une restriction comme une règle locale suivie d'une règle plus large mais moins prioritaire. On a pour une infirmière :

r1 : interdiction sur DAET2 ;

r2 : autorisation sur DAET ;

r3 : interdiction sur DA ;

r4 : autorisation sur DAES.

On obtient en triant les règles dans l'ordre défini par $\leq$, dans le cas d'une requête sur l'épisode E de l'établissement A :

$$r1 \leq r2 \leq r4 \leq r3$$

Le parcours des règles 1 à 4 par l'algorithme 2 donne le résultat suivant :

r1 $int_i = DAET2$;

r2 $aut_i = DAET \setminus DAET2$;

r4 $aut_i = (DAET \setminus DAET2) \cup (DAES \setminus DAET2)$;

r3 $int_i = DAET2 \cup DA = DA$.

Les données accessibles sont donc :

$aut = (DAET \setminus DAET2) \cup (DAES \setminus DAET2)$;

En modifiant le critère qui rend une règle applicable, on peut construire avec l'algorithme 2 l'ensemble des données accessibles depuis un profil : il suffirait d'enlever la clause $cible \leq r.cible$. Notons que si l'algorithme est appliqué à un profil et non à une personne, les permissions/interdictions particulières à cette personne seront omises. En outre, non seulement on peut construire l'ensemble des données auxquelles a accès une personne, mais on peut également construire l'ensemble des profils qui ont accès à une donnée sur le même principe.

Chapitre 5

Cas d'étude

Ce chapitre illustre l'algorithme décrit dans le chapitre 4. Pour cet exemple, un patient est transféré depuis Magog vers le CHUS, car une analyse de sang a révélé une anomalie qui nécessite une intervention qui n'est pas possible au sein de l'infrastructure de Magog. Ses données sont représentées par la hiérarchie de données de la figure 5.2.

Le patient ayant son beau-frère travaillant au CHUS, dont une partie de la hiérarchie est présente sur la figure 5.1, n'autorise que lui parmi l'ensemble des chirurgiens anesthésistes à avoir accès à la donnée Opération de son ancienne opération et interdit également l'ensemble des chirurgiens anesthésistes d'avoir accès aux données issues du bloc opératoire. Afin d'être en mesure de recevoir des soins, il permet tout de même aux chirurgiens de n'avoir accès qu'à Episode Operation privé de Opération parmi les données du CHUS et de Magog.

On a donc au niveau des règles du consentement explicite :

- une *permission* pour le beau-frère d'accéder à Opération ;
- une *interdiction* pour le profil Chirurgien Anesthésiste d'accéder à Bloc opératoire ;
- une *restriction de permission* pour le profil Chirurgien sur Episode Operation privé de Opération parmi *les données du CHUS et de Magog*.

En outre, dans le cas d'un transfert inter-établissement, le patient consent implicitement à ce que le personnel soignant de l'établissement d'arrivée qui s'occupe de lui ait accès aux données du dernier épisode de l'établissement d'origine. Cependant, l'hôpital ayant eu des problèmes dans la gestion des règles d'accès, interdit aux

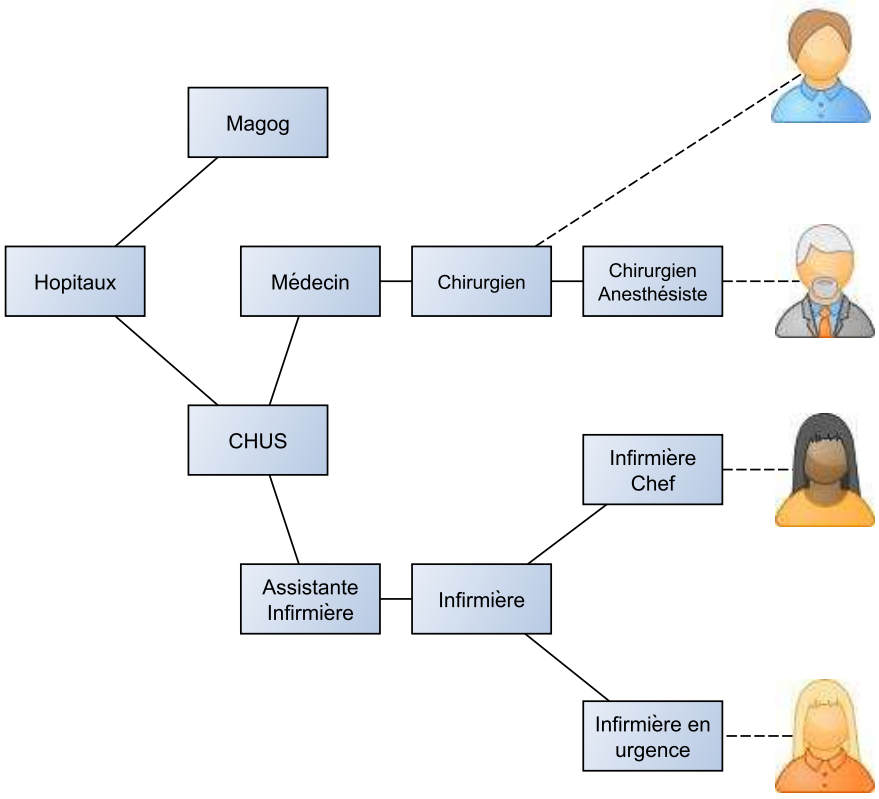


Figure 5.1 – Cas d'étude : hiérarchie de profils

médecins la consultation des parties concernant les soins post-opératoires sans l'aval exprès du patient.

Ce qui donne après ré-écriture de la restriction, pour la requête <accéder(beau-frère,DMP)> :

- au niveau des règles explicites,
  - r1 : permission pour le sujet beau-frère sur la cible Opération,
  - r2 : interdiction pour le sujet Chirurgien Anesthésiste sur la cible Bloc opératoire,
  - r3 : permission pour le sujet Chirurgien sur la cible Episode Opération Pied privé de Opération,
  - r4 : interdiction pour le sujetChirurgien sur la cible CHUS ,
  - r5 : interdiction pour le sujetChirurgien sur la cible Magog ;

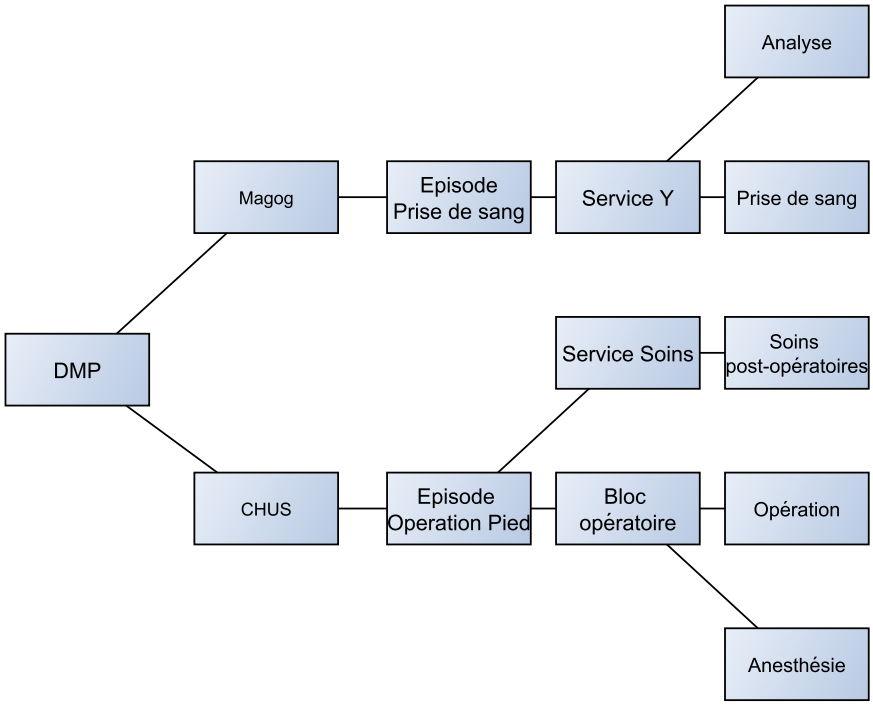


Figure 5.2 – Cas d'étude : hiérarchie de données du patient

– au niveau des règles implicites,

r6 : permission pour le sujet CHUS pour sur la cible Episode prise de sang,

r7 : interdiction pour le sujet Médecin sur la cible Soins post-opératoires.

Si le beau-frère demande tout l'épisode Episode Operation Pied, les règles applicables sont r1, r2, r3, r4, r5, r6 et r7. L'ordre de traitement sera r1, r2, r3, r4,r5 et r6, r7.

On aurait donc pour le niveau 1 :

r1 : $aut_1 = Opération$;

r2 : $int_1 = Bloc\_opératoire$;

r3 : $aut_1 = Opération \cup ((Episode\_Opération\_Pied \setminus Opération) \setminus Bloc\_opératoire)$
$= Opération \cup (Episode\_Opération\_Pied \setminus Bloc\_opératoire)$;

r4 : $int_1 = Bloc\_opératoire \cup CHUS$;

r5 : $int_1 = Bloc\_opératoire \cup CHUS \cup Magog$;

Puis pour le niveau 2 :

r6 : $aut_2 = Episode\_prise\_de\_sang$;
r7 : $int_2 = Soins\_postopératoires$;
Enfin :

$$aut = aut_1 \cup (aut_2 \setminus int_1) = aut_1;$$

La figure 5.3 montre quelles sont les données accessibles par le beau-frère en prenant en compte le consentement explicite (cadres entourés par des tirets) et le consentement implicite (cadres entourés par des pointillés). L'ensemble des données accessible est en vert, l'ensemble interdit en rouge.

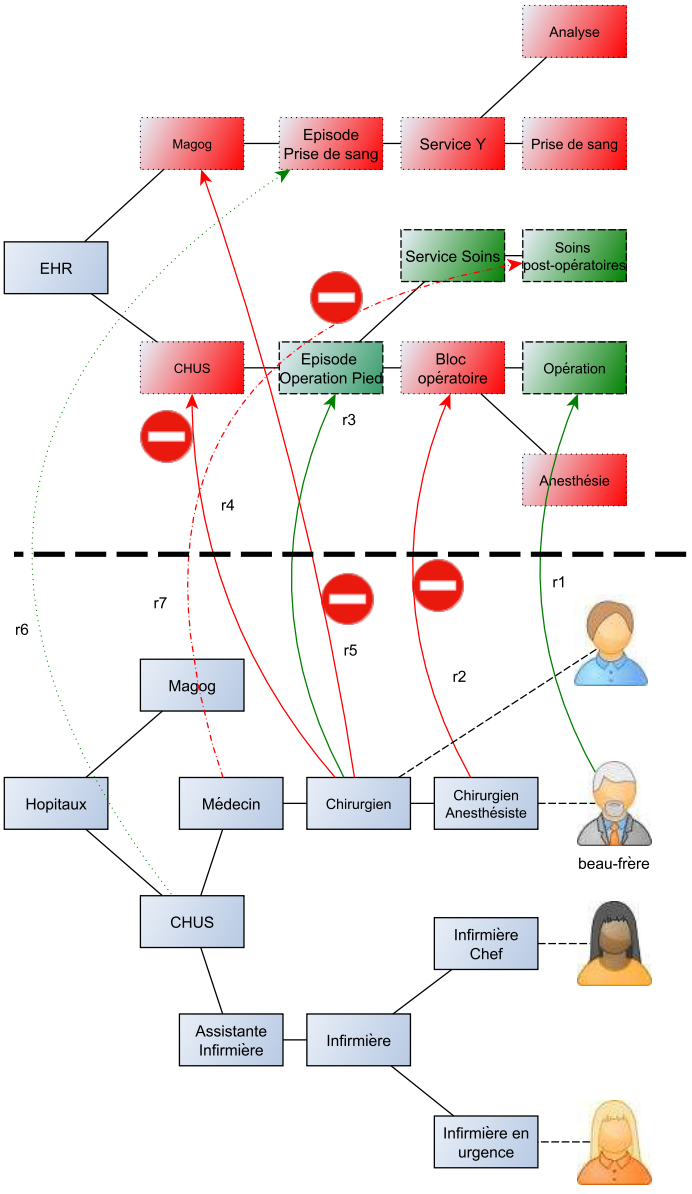


Figure 5.3 – Cas d'étude : données accessibles par le sujet

Conclusion

On a donc vu que la prise en compte du consentement du patient bouleverse les contrôles d'accès habituels tels que RBAC. La première partie du mémoire a bien montré que les différentes solutions utilisées sont statiques : que ce soit l'opt-in partiel, l'opt-out partiel ou encore l'opt-in complet, il n'est pas possible de spécifier des règles dynamiques qui peuvent être valables sous certaines conditions. En effet, il n'est possible de spécifier qu'un comportement statique du système avec les solutions en vigueur dans les états étudiés.

Nous avons étudié des solutions de gestion dynamique de contrôle d'accès afin de créer notre propre solution qui s'adapterait à notre problème, à savoir gérer le consentement dans les accès aux dossiers médicaux électroniques. Le sujet ayant des répercussions légales, il a fallu étudier quels types d'informations sont diffusés et sous quelles conditions les accès sont permis ou interdits. Pour cela, nous avons collaboré avec des gens du CHUS et plus particulièrement avec des archivistes, chargés de veiller sur les dossiers médicaux papier. Les deux solutions les plus prometteuses étant Ponder et Cassandra, nous avons effectué des études plus poussées afin de déterminer les avantages/inconvénients.

Fort de ces résultats, nous avons mis sur pied un prototype dont l'implémentation a été effectuée par Thierry Le Ba, basé sur les points forts des solutions précédentes. Nous avons fourni un algorithme de gestion dynamique des conflits afin de déterminer à partir d'une requête <action(sujet,cible)> quelles données seront accessibles au sujet. Un des points forts de notre algorithme est qu'il est possible à partir d'une cible (une donnée donc) de savoir quels sujets peuvent y avoir accès et ainsi il est possible de faire de la vérification de propriétés telles que «le patient X peut toujours être traité par le médecin M avec les règles d'accès qu'il a instauré».

Les différentes entrevues avec le personnel du CHUS n'ont pas permis de valider entièrement l'algorithme de gestion de conflit : pour l'instant, les patients sont trop peu habitués à la possibilité d'exprimer leur consentement dans les accès à leur dossier. En effet la plupart des demandes que reçoivent les archivistes du CHUS ne sont que des vérifications pour savoir si une personne a consulté les dossiers du patient. Il y a donc très peu de cas de conflits pour valider l'algorithme mais les différents scénarii imaginés valident l'algorithme partiellement. Une des préconditions pour que l'algorithme fonctionne est d'avoir une hiérarchie de profil : il faut pouvoir construire un arbre de profil avec la condition que chaque profil fils hérite des «permissions» du profil père. Cette précondition constitue pour l'instant le défaut majeur car elle peut s'avérer être difficile à réaliser.

Les futurs travaux pourront porter sur le problème des hiérarchies, qui se rapporte en fait à la recherche d'une relation d'ordre entre les règles de consentement. En outre, la possibilité de faire de la vérification sur les accès est un atout majeur pour sensibiliser le patient aux choix qu'il fait et le prévenir que dans certains cas, il ne pourra pas être traité convenablement. Cela constituera un axe intéressant de recherche qui bénéficiera grandement au projet en commun avec le CHUS.

Annexe A

Code JAVA des classes Patient et Carer

programme A.1 – Code de la classe Patient

```
package net.ponder2.managedobject;

import net.ponder2.ManagedObject;
import net.ponder2.apt.Ponder2op;
import net.ponder2.exception.Ponder2ArgumentException;
import net.ponder2.exception.Ponder2Exception;
import net.ponder2.exception.Ponder2OperationException;
import net.ponder2.objects.P2Object;

public class Patient implements ManagedObject{

        private String name;
        private P2Object carer;
        private P2Object metarecord;
        private P2Object myP2Object;
        private String organization;
          /*
           * create an instance of Patient
           * @param nom : name of the patient
           * @param myP2object : automatically given when this method
           * is called from Pondertalk
```

```
 */
@Ponder2op("create:")
public Patient(P2Object myP2Object, String nom){
        this.myP2Object = myP2Object;
        this.name = nom;
}

 /*
 * create an instance of Patient with an organization
 * @param nom : name of the patient
 * @param myP2object : automatically given when this method
 * is called from Pondertalk
 * @param org : name of the organization
 */
@Ponder2op("create:org:")
public Patient(P2Object myP2Object,
                        String nom, String org){
        this.myP2Object = myP2Object;
        this.name = nom;
        this.organization = org;
}

/*
 * access to patient's data (dumb test)
 */
@Ponder2op("data")
public void data(){
      System.out.println("accessing "+this.name+"'s data");
}

@Ponder2op("getName")
public String getName(){
return name;
}

@Ponder2op("setCarer:")
public void setCarer(P2Object carer) throws
Ponder2ArgumentException,
Ponder2OperationException,
```

```
Ponder2Exception{
        String name = carer.operation(myP2Object,"getName")
              .asString();
        System.out.println("patient "+this.name
              +" is treated by " + name);
        this.carer = carer;
}

@Ponder2op("getCarer")
public P2Object getCarer(){
        return carer;
}

/*
 * Check if carer is the patient's carer
 */
@Ponder2op("HasCarer:")
public boolean HasCarer(P2Object carer){
        return carer == this.carer;
}

@Ponder2op("getRecord")
public P2Object p2_operation_getRecord(){
        return this.metarecord;
}

@Ponder2op("setRecord:")
public void p2_operation_setRecord(P2Object record){
        this.metarecord = record;
}

/*
 *  retrieve last episode from this patient
 *  @param organization
 *  @return the last episode
 */

@Ponder2op("getLastEpisode:")
public P2Object p2_operation_getLastEpisode(
```

```
        String organization) throws Ponder2Exception{
          P2Object record = metarecord.operation(metarecord,
                        "at:",organization);
          P2Object episode = record.operation(record, "getLast");
          System.out.println("accessing␣"+ name
                        +"'s␣last␣episode");
          String datastring = episode.operation(episode,
                        "consult").asString();
          System.out.println("last␣episode:␣"+datastring);
          return episode;
}

/*
 *  retrieve a past episode from this patient
 *  @param index of the wanted episode
 *  @param organization
 *  @return the past episode
 */

@Ponder2op("getPastEpisode:org:")
public P2Object p2_operation_getPastEpisode(int index,
                String organization) throws
                Ponder2Exception{
          P2Object record = metarecord.operation(metarecord,
                "at:",organization);
          P2Object episode = record.operation(record, "get:",
                String.valueOf(index));
          System.out.println("accessing␣"+ name
                +"'s␣past␣episode␣"+index);
          String datastring = episode.operation(episode,
                "consult").asString();
          System.out.println("past␣episode␣"+index+":␣"+datastring);
          return episode;
}



/*
 *  retrieve the last episode from a patient from an organization
```

```
 *  @param patient
 *  @param organization : name of the organization
 *  @return the last episode
 */
@Ponder2op("accessLastEpisode:org:")
public P2Object p2_operation_accessLastEpisode(
P2Object patient,String organization)
throws Ponder2ArgumentException,
Ponder2OperationException,
Ponder2Exception{
        System.out.println("p_subject: " +myP2Object);
        System.out.println("p_target: "+ patient);
        String name = patient.operation(myP2Object,"getName")
              .asString();
        System.out.println(this.name + ": trying to acces "
                           + name + "'s last episode"
                           );
        P2Object data = patient.operation(myP2Object,
              "getLastEpisode:",
              organization);
        return data;
}
/*
 *  retrieve a past episode from a patient
 *  from an organization
 *  @param index of the wanted episode
 *  @param organization
 *  @return the past episode
 */

@Ponder2op("accessPastEpisode:org:ep:")
public P2Object p2_operation_accessPastEpisode(
      P2Object patient,String organization,int index) throws
      Ponder2ArgumentException,
      Ponder2OperationException,
      Ponder2Exception{
        System.out.println(myP2Object);
        System.out.println(patient);
        String name = patient.operation(myP2Object,"getName")
```

```
                .asString();
            System.out.println(this.name + ": trying to acces "
                                + name + "'s past episode"
                                );
            P2Object data = patient.operation(myP2Object,
                "getPastEpisode:org:",String.valueOf(index),
                organization);
            return data;
        }

        @Ponder2op("getOrganization")
        public String p2_operation_getOrganization(){
            return organization;
        }


        @Ponder2op("setOrganization:")
        public void p2_operation_setOrganization(String org){
            organization = org;
        }

}
```

programme A.2 – Code de la classe Carer

```
package net.ponder2.managedobject;

import net.ponder2.ManagedObject;
import net.ponder2.apt.Ponder2op;
import net.ponder2.exception.Ponder2ArgumentException;
import net.ponder2.exception.Ponder2Exception;
import net.ponder2.exception.Ponder2OperationException;
import net.ponder2.objects.P2Object;

public class Carer implements ManagedObject{
        private String name;
        private P2Object myP2Object ;
        private String organization;


        @Ponder2op("create:")
        public Carer(P2Object myP2Object,String name){
                this.myP2Object = myP2Object;
                this.name=name;
        }

        @Ponder2op("create:org:")
        public Carer(P2Object myP2Object,String name,String org){
                this.myP2Object = myP2Object;
                this.name=name;
                organization = org;
        }

        @Ponder2op("access:")
        public P2Object access(P2Object patient) throws
        Ponder2ArgumentException,
        Ponder2OperationException,
        Ponder2Exception{
                String name = patient.operation(myP2Object,"getName")
                                .asString();
                System.out.println(this.name + ":␣trying␣to␣acces␣"
                                + name + "'s␣data"
```

```
                        );
        P2Object data = patient.operation(myP2Object,"data");
return data;
}

  @Ponder2op("getName")
  public String getName(){
  return name;
  }

  @Ponder2op("accessLastEpisode:org:")
  public P2Object p2_operation_accessLastEpisode(
  P2Object patient,String organization)
  throws Ponder2ArgumentException,
  Ponder2OperationException,
  Ponder2Exception{
          String name = patient.operation(myP2Object,"getName")
                        .asString();
          System.out.println(this.name + ":␣trying␣to␣acces␣"
                        + name + "'s␣last␣episode"
                        );
          P2Object data = patient.operation(myP2Object,
                        "getLastEpisode:",organization);
          return data;
  }


  @Ponder2op("accessPastEpisode:org:ep:")
  public P2Object p2_operation_accessPastEpisode(
  P2Object patient,String organization,int index)
  throws Ponder2ArgumentException,
  Ponder2OperationException,
  Ponder2Exception{
          String name = patient.operation(myP2Object,"getName")
                        .asString();
          System.out.println(this.name + ":␣trying␣to␣acces␣"
                        + name + "'s␣past␣episode"
                        );
          P2Object data = patient.operation(myP2Object,
```

```
                                "getPastEpisode:org:",
                                String.valueOf(index),
                                organization);
                return data;
        }

        @Ponder2op("getOrganization")
        public String p2_operation_getOrganization(){
                return organization;
        }

        @Ponder2op("setOrganization:")
        public void p2_operation_settOrganization(String org){
                organization=org;
        }

        @Ponder2op("equal:")
        public boolean p2_operation_equal(P2Object obj){
                return obj == myP2Object;
        }
}
```

Annexe A. Code JAVA des classes Patient et Carer

Annexe B

Code PonderTalk

programme B.1 – Fonctions PonderTalk

```
//Fonction.p2
domain := root/factory/domain.

//chargement des classes JAVA
//et mise en place des constructeur dans la hierarchie
patientFactory := root load: "Patient".
root/factory at: "patient" put: patientFactory.

carerFactory := root load: "Carer".
root/factory at: "carer" put: carerFactory.

recordFactory := root load: "Record".
root/factory at: "record" put: recordFactory.

episodeFactory := root load: "Episode".
root/factory at: "episode" put: episodeFactory.

metarecordFactory := root load: "Metarecord".
root/factory at: "metarecord" put: metarecordFactory.



//creation d'une fonction d'ajout d'hopital
```

```
addhospital := [ :name :services |
    hospital := root/hospital.
        hospital := root/hospital at: name put: domain create.

        // Creations des services et du personnel
        hospital at: "ward" put: domain create.
        hospital at: "personnel" put: domain create.

        services do: [ :name |
                ward := hospital/ward at: name put: domain create.
                ward at: "bed" put: domain create.
                ward at: "personnel" put: domain create.
        ].
].

root/sc at: "addHospital" put: addhospital.


//creation d'une fonction d'ajout de patient

addpatient := [ :hospital :ward :name |
        //recuperation du domaine de l'hopital
        hdom := root/hospital at: hospital.
        //recuperation du domaine du service
        sdom := hdom/ward at: ward.
        //recuperation du domaine du lit
        ldom := sdom/bed.
        //creation du patient
        pat := root/factory/patient create: name org: hospital.
        patient := ldom at: name put: pat.
        //creation du contenu du patient
        record := root/factory/metarecord create.
        patient setRecord: record.

].

root/sc at: "addPatient" put: addpatient.
```

```
//creation d'une fonction d'ajout d'un intervenant

addcarer := [ :hospital :ward :name |
        //recuperation du domaine de l'hopital
        hdom := root/hospital at: hospital.
        //recuperation du domaine du service
        sdom := hdom/ward at: ward.
        pdom2 := hdom at: "personnel".
        //recuperation du domaine du personnel
        pdom := sdom/personnel.
        //creation de l'intervenant
        c := root/factory/carer create: name org: hospital.
        pdom at: name put: c.
        pdom2 at: name put: c.
].

root/sc at: "addCarer" put: addcarer.

getPatient := [ :hospital :ward :name |
        //recuperation du domaine de l'hopital
        hdom := root/hospital at: hospital.
        //recuperation du domaine du service
        sdom := hdom/ward.
        sdom2 := sdom at: ward.
        sdom3 := sdom2/bed.
        sdom3 at: name.
].

root/sc at: "getPatient" put: getPatient.

getCarer := [ :hospital :name |
        //recuperation du domaine de l'hopital
        hdom := root/hospital at: hospital.
        //recuperation de l'intervenant
        sdom := hdom/personnel.
        sdom at: name.
].
```

```
root/sc at: "getCarer" put: getCarer.


//creation d'une fonction de deplacement d'un patient

movepatient := [ :hopitald_service_dom :hopitala_service_dom :patient |
        bdomd := hopitald_service_dom at: "bed".
        bdoma := hopitala_service_dom at: "bed".
//      patient := bdomd at: name
                ifAbsent: [root print: "patient introuvable"].
        name := patient getName.
        [bdoma at: name put: patient]
                whileTrue: [ bdomd removeObject: patient ].
        [bdomd removeObject: patient]
                whileTrue: [ bdomd removeObject: patient ].
].

root/sc at: "movePatient" put: movepatient.
```

programme B.2 – Règles PonderTalk

```
//-------------------------------------------------------------
//premiers tests avec la fonction data
//-------------------------------------------------------------

//interdiction d'acceder au dossier medical d'un patient

newauthpol := root/factory at: "authpolicy".

root/policy at: "global_deny_data" put:
                        (newauthpol subject: root
                                                action: "data"
                                                target: root
                                                focus: "s").
root/policy/global_deny_data reqneg.
root/policy/global_deny_data active: true.



//autorisation pour un intervenant d'acceder au dossier d'un
//patient s'il est son intervenant traitant

root/policy at: "rule_traiting_patient" put:
                        (newauthpol subject: root
                                                action: "data"
                                                target: root
                                                focus: "s").
root/policy/rule_traiting_patient reqcondition:
                [p_target HasCarer: p_subject].
root/policy/rule_traiting_patient final.
root/policy/rule_traiting_patient active: true.

//-------------------------------------------------------------
//-Auth deny fonctionne mal ?
//Mise en place d'interdictions globales
//+autorisation specifiques
//-------------------------------------------------------------
```

```
// interdiction sur les actions
// getLastEpisode, getName, getPastEpisode

root/policy at: "global_deny_getLastEpisode" put:
                        (newauthpol subject: root
                                                action: "getLastEpisode:"
                                                target: root
                                                focus: "s").
root/policy/global_deny_getLastEpisode  reqneg.
root/policy/global_deny_getLastEpisode active: true.

root/policy at: "global_deny_getName" put:
                        (newauthpol subject: root
                                                action: "getName"
                                                target: root
                                                focus: "s").
root/policy/global_deny_getName reqneg.
root/policy/global_deny_getName active: true.

root/policy at: "global_deny_getPastEpisode" put:
                        (newauthpol subject: root
                                                action: "getPastEpisode:org:"
                                                target: root
                                                focus: "s").
root/policy/global_deny_getPastEpisode  reqneg.
root/policy/global_deny_getPastEpisode active: true.


//-------------------------------------------------------------
//autorisations specifiques
//-------------------------------------------------------------


//autorisation d'acceder a son propre dossier
//la regle n'a pas besoin d'etre marquee finale car elle est
//plus specifique que l'interdiction

root/policy at: "global_auth_getLastEpisode_self" put:
                        (newauthpol subject: root/hospital
```

```
                                                action: "getLastEpisode:"
                                                target: root/hospital
                                                focus: "s").
root/policy/global_auth_getLastEpisode_self
                        reqcondition: [p_target == p_subject].
root/policy/global_auth_getLastEpisode_self active: true.
//root/policy/global_auth_getLastEpisode_self final.

// autorisation d'acceder a son propre dossier
// la regle n'a pas besoin d'etre marquee finale car elle est
// plus specifique que l'interdiction

root/policy at: "global_auth_getPastEpisode_self" put:
                        (newauthpol subject: root/hospital
                                                action: "getPastEpisode:org:"
                                                target: root/hospital
                                                focus: "s").
root/policy/global_auth_getLastEpisode_self
                reqcondition: [p_target == p_subject].
root/policy/global_auth_getLastEpisode_self active: true.
//root/policy/global_auth_getLastEpisode_self final.


root/policy at: "global_auth_getName_self" put:
                        (newauthpol subject: root/hospital
                                                action: "getName"
                                                target: root/hospital
                                                focus: "s").
//root/policy/global_auth_getName_self
                reqcondition: [p_target == p_subject].
root/policy/global_auth_getName_self active: true.
//root/policy/global_auth_getName_self final.

// autorisation pour un medecin traitant d'acceder au dossier de son patient

// un medecin traitant peut avoir acces au dernier episode
// de l'etablissement du patient
// soit dans le sien (cas courant)
// soit dans l'etablissement d'origine du patient (cas du transfert)
```

```
root/policy at: "global_auth_getLastEpisode_carer_CHUS" put:
                (newauthpol subject: root/hospital/CHUS
                                    action: "getLastEpisode:"
                                    target: root/hospital/CHUS/ward
                                    focus: "s").

root/policy/global_auth_getLastEpisode_carer_CHUS
                reqcondition: [:getLastEpisode|
                [getLastEpisode == p_subject getOrganization] value
                and: [ p_target HasCarer: p_subject]].
root/policy/global_auth_getLastEpisode_carer_CHUS active: true.


root/policy at: "global_auth_getLastEpisode_carer_CLSC" put:
                (newauthpol subject: root/hospital/CLSC
                                    action: "getLastEpisode:"
                                    target: root/hospital/CLSC
                                    focus: "s").
root/policy/global_auth_getLastEpisode_carer_CLSC
                reqcondition: [:getLastEpisode|
                getLastEpisode == p_subject getOrganization
                and: [p_target HasCarer: p_subject]].
root/policy/global_auth_getLastEpisode_carer_CLSC active: true.

root/policy at: "global_auth_getLastEpisode_carer_CHUS_transfert" put:
                (newauthpol subject: root/hospital/CHUS
                                    action: "getLastEpisode:"
                                    target: root/hospital/CHUS/ward
                                    focus: "s").
root/policy/global_auth_getLastEpisode_carer_CHUS_transfert
                        reqcondition: [:getLastEpisode|
                        [getLastEpisode == p_target getOrganization] value
                        and: [ p_target HasCarer: p_subject]].
root/policy/global_auth_getLastEpisode_carer_CHUS_transfert active: true.

root/policy at: "global_auth_getLastEpisode_carer_CLSC_transfert" put:
                (newauthpol subject: root/hospital/CLSC
                                    action: "getLastEpisode:"
```

```
                                        target: root/hospital/CLSC/ward
                                        focus: "s").
root/policy/global_auth_getLastEpisode_carer_CLSC_transfert
                        reqcondition: [:getLastEpisode|
                        [getLastEpisode == p_target getOrganization] value &
                        [ p_target HasCarer: p_subject] value
                        & [p_target getOrganization != p_subject
                        getOrganization] value].
root/policy/global_auth_getLastEpisode_carer_CLSC_transfert
                        active: true.
```

Annexe B. Code PonderTalk

Bibliographie

[1] Joachim Bergmann, Oliver J. Bott, Dietrich Peter Pretschner et Reinhold Haux.
« An e-consent-based shared EHR system architecture for integrated healthcare networks ».
I. J. Medical Informatics, 76(2-3):130–136, 2007.

[2] Jeremy Bryans.
« Reasoning about XACML policies using CSP ».
Dans Ernesto Damiani et Hiroshi Maruyama, éditeurs, *SWS*, pages 28–35. ACM, 2005.

[3] Enrico Coiera et Roger Clarke.
« e-Consent : The Design and Implementation of Consumer Consent Mechanisms in an Electronic Environment ».
Journal of the American Medical Informatics Association, 11(2):129–140, Mars/April 2004.

[4] Nicodemos Damianou, Naranker Dulay, Emil Lupu et Morris Sloman.
« Ponder : A Language for Specifying Security and Management Policies for Distributed Systems », 2000.

[5] Nicodemos Damianou, Naranker Dulay, Emil Lupu et Morris Sloman.
« The Ponder Policy Specification Language ».
Dans Morris Sloman, Jorge Lobo et Emil Lupu, éditeurs, *POLICY*, volume 1995 de *Lecture Notes in Computer Science*, pages 18–38. Springer, 2001.

[6] Gouvernement du Québec.
« Loi sur l'accès aux documents des organismes publics et sur la protection des

renseignements personnels ».
Lois refondues du Québec, chapitre A-2.1, À jour au 1er septembre 2011.

[7] Gouvernement du Québec.
« Loi sur les services de santé et les services sociaux ».
Lois refondues du Québec, chapitre S-4.2, À jour au 1er septembre 2011.

[8] Loi du Québec.
« Code civil du Québec ».

[9] D. Ferraiolo, D.R. Kuhn et R. Chandramouli.
Role-based access control.
Artech House computer security series. Artech House, 2003.

[10] Melissa M. Goldstein et Alison L. Rein.
« *Consumer Consent Options for Electronic Health Information Exchange : Policy Considerations and Analysis* ».
The Office of the National Coodinator for Health Information Technology, 23 Mars 2010.

[11] C. A. R. Hoare.
« Communicating Sequential Processes (Reprint) ».
Commun. ACM, 26(1):100–106, 1983.

[12] Jing Jin, Gail-Joon Ahn, Hongxin Hu, Michael J. Covington et Xinwen Zhang.
« Patient-centric authorization framework for electronic healthcare services ».
Computers & Security, 30(2-3):116–127, 2011.

[13] Wilf R. LaLonde.
Discovering smalltalk.
Benjamin/Cummings series in object-oriented software engineering. Benjamin/-Cummings, 1994.

[14] J Longstaff.
« Messages and Overrides : enhancements to Sealed Envelope authorisation ».
Dans *HC2009*. Health Informatics Forum of the British Computer Society, 2009.

[15] Marco Casassa Mont et Robert Thyne.
« Privacy policy enforcement in enterprises with identity management solu-

tions ».
Dans *PST*, volume 380 de *ACM International Conference Proceeding Series*, page 25. ACM, 2006.

[16] Huda M. N., Yamada, S. et N. Sonehara.
« Privacy-aware Access to Patient-controlled Personal Health Records in Emergency Situations ».
3rd International Conference on Pervasive Computing Technologies for Healthcare, May 2009.

[17] Joy Pritts et Kathleen Connor.
« *The Implementation of E-consent Mechanisms in Three Countries : Canada, England, and the Netherlands* ».
The Substance Abuse and Mental Health Services Administration of the U.S. Department of Health and Human Services, 16 Février 2017.

[18] Giovanni Russello, Changyu Dong et Naranker Dulay.
« Authorisation and Conflict Resolution for Hierarchical Domains ».
Dans *POLICY*, pages 201–210. IEEE Computer Society, 2007.

[19] Giovanni Russello, Changyu Dong et Naranker Dulay.
« Consent-Based Workflows for Healthcare Management ».
Dans *POLICY*, pages 153–161. IEEE Computer Society, 2008.

[20] Nan Zhang, Mark Ryan et Dimitar P. Guelev.
« Synthesising verified access control systems through model checking ».
Journal of Computer Security, 16(1):1–61, 2008.