THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Evaluation automatique des politiques de sécurité

Tournié, Stéphane

Award date:
2012

Awarding institution:
Universite de Namur

Facultés Universitaires
Notre-Dame de la Paix de Namur

Faculté d'Informatique

Évaluation automatique des politiques de sécurité

Stéphane Tournié

Mémoire présenté en vue de l'obtention du grade de Master en informatique

Professeur : Jean-Noël Colin

Année académique : 2011-2012

Résumé

Différents langages permettent de définir des politiques de sécurité, plus particulièrement des politiques de contrôle d'accès. Pour être efficaces, ces politiques doivent pouvoir être évaluées en temps réel au moment où un accès est demandé. Cela nécessite un outil performant de prise de décision sur base de règles et de faits existants. Ce travail tente d'identifier les technologies aptes à supporter de manière efficace l'évaluation de politiques et à concevoir et développer un prototype pour le moteur d'évaluation de règles basé sur un coeur d'évaluation en XACML. De plus, le prototype est conçu de manière modulaire et extensible afin de permettre le support de différents langages et l'ajout du support de nouveaux types de règles par la suite.

Mots clés: XACML, REL, moteur d'évaluation de règles.

Abstract

Several langagues can be used to define security policies and in fact access control policies. To be consistent, such policies have to be evaluated in real time when a request for access occurs. To take decisions, based on the rules and facts from a base of knowledge, it needs a suitable and performant system. This work try to identify technologies capable to support effectively the evaluation of policies and to design and build a prototype for the rules evaluation engine based on a core evaluation module in XACML. Moreover, the prototype is designed in a modular and extensible way in order to accept several policy expression languages and to give the opportunity to add support of new policies types afterthought.

Keywords: XACML, REL, rules evaluation engine.

Je tiens à remercier particulièrement mon promoteur Jean-Noël Colin pour son support, ses relectures, ses conseils et sa disponibilité.

Un grand merci à Christelle Remy et Stéphanie Rouillard pour leurs relectures attentives.

Je tiens à remercier toutes les personnes qui m'ont soutenues, de près ou de loin, durant toute la durée de la réalisation de ce mémoire, et en particulier mes parents pour leurs conseils et leur présence, toujours positive.

À tous, merci.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ACRONYMES

DAC	Discretionary Access Control
DRM	Digital Rights Management
EPAL	Enterprise Privacy Authorization Language
MAC	Mandatory Access Control
MPEG-21 REL	MPEG-21 Rights Expression Language
ODRL	Open Digital Rights Language
OrBAC	Organisation-Based Access Control
P-RBAC	Privacy-aware Role Based Access Control
R-BAC	Role-Based Access Control
REL	Rights Expression Language
SI	Système d'Information
URI	Uniform Resource Identifier
URL	Uniform Resource Link
URN	Uniform Resource Name
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

CHAPITRE 1

INTRODUCTION

Les systèmes informatiques actuels, de plus en plus divers et reliés par des réseaux toujours plus nombreux et variés, proposent aux utilisateurs des dizaines de milliers de services de tout type. En 2012, un système informatique prend tout autant la forme d'un smartphone, exécutant des applications mobiles, que d'une architecture d'entreprise, déployée sur des dizaines de postes de travail et offrant à ses utilisateurs des outils partagés de bureautique, de modélisation de processus ou des capacités de calcul élevées. Dans un tel contexte de diversité des ressources informatiques et des utilisateurs des systèmes, la nécessité de maîtriser parfaitement ce que chaque utilisateur (ou système automatisé) possède le droit de faire et de ne pas faire est primordiale.

Plus concrétement, il est nécessaire, au sein de chaque système informatique, de contrôler les accès des utilisateurs et systèmes automatisés aux ressources offertes. Étant donné la variété des types de ressources à prendre en compte ainsi que la multitude des contextes d'utilisation de celles-ci, des politiques complètes sont définies pour le contrôle des accès aux systèmes, favorisant la formalisation de méthodes à appliquer dans ce cadre.

1.1 Problématique

Des technologies supportant la définition de politiques de sécurité existent pour répondre aux exigences de domaines spécifiques tels que le contrôle de l'accès physique à des ressources, la gestion des droits liés à des contenus numériques ou l'établissement de permissions pour des systèmes de fichier. La difficulté principale réside dans l'interopérabilité et la communication entre des politiques provenant de domaines différents afin de constituer une sécurité cohérente d'un système com-

plexe.

Une fois qu'une politique et principalement les règles qui la composent ont été définies pour un système, se pose la question de l'évaluation de celles-ci, en vue de répondre à des demandes d'accès. Ce mémoire se focalise sur l'**évaluation automatique** des politiques, qui est nécessaire pour rendre une décision d'autorisation en temps réel et efficace.

Cette nécessité d'évaluer des politiques en temps réel, de manière automatique, demande d'avoir recours à un outil de prise de décision adéquat. L'éventail des langages permettant d'exprimer des politiques de sécurité étant très large, se pose la question d'une part, du choix des modes de représentations qu'il est important de considérer et d'autre part, de l'intégration de mécanismes d'évaluation de ces différents langages en un outil efficace.

C'est au prix de la construction d'un tel outil suffisamment modulaire et générique que le décloisonnement des politiques de sécurité provenant de différents environnements et systèmes pourra être envisagé. En effet, vu l'étendue des cas d'utilisation et la difficulté d'imposer un nombre de standards limité, recourir à des technologies diverses empêche trop souvent les échanges entre domaines d'application.

De plus, les standards et langages évoluent tout autant que les usages et des technologies définies aujourd'hui uniquement pour un champ d'actions particulier ne survivront pas aux changements. La construction d'un outil d'évaluation de politiques doit donc également veiller à rester ouvert et extensible a posteriori.

1.2 Questions de recherche

Différentes priorités de travail se dégagent de ces problèmatiques et permettent de préciser l'axe à suivre. Tout d'abord, il apparaît nécessaire d'identifier les principaux modèles et langages permettant de représenter des politiques de sécurité.

Ensuite, les exigences principales pour la construction d'un moteur d'évaluation générique ressortent assez clairement.

Nous posons ici trois questions de recherche centrales. Celles-ci ouvrent donc un questionnement à différents niveaux, visant à la fois à explorer le contexte et les solutions déjà existantes et à dessiner les caractéristiques d'une architecture d'évaluation de politiques de sécurité générique, extensible et efficace :

Q1 Quelle est la meilleure manière de définir des règles de sécurité et d'exprimer des contraintes ?

Q2 Quelles technologies sont aptes à supporter de manière efficace l'évaluation automatique de politiques ?

Q3 Comment adopter une approche générique dans la construction d'une architecture d'évaluation automatique de politiques ?

1.3 Structure du document

Pour répondre à ces questions, nous commençons par une analyse des différents aspects du contexte dans lequel ce travail s'inscrit. Dans le **Chapitre 2** sont détaillés les principaux modèles de contrôle d'accès qui, selon les contraintes propres d'une organisation, sont susceptibles d'être utilisés comme cadre pour la définition de règles de contrôle d'accès. Le **Chapitre 3** décrit les langages de description de politiques majeurs qui permettent d'exprimer des règles de contrôle d'accès. Les caractéristiques des deux principaux langages de description de politiques sont détaillées alors que dans un **Chapitre 4**, les principaux langages d'expression de droits numériques sont présentés. Introduisant la problèmatique d'une approche modulaire pour la mise en place d'un prototype de moteur d'évaluation de règles, une attention particulière est ensuite portée aux tentatives déjà explorées d'intéroperabilité entre de tels langages.

Dans une seconde partie de ce travail, est repris l'ensemble des éléments ayant per-

mis de réaliser ou faisant partie du prototype de moteur d'évaluation de règles qui constitue le coeur de ce mémoire. Ainsi, les éléments de conception du prototype sont exposés au **Chapitre 5** alors que le **Chapitre 6** explique le fonctionnement du système qui a été dessiné, des choix demandés lors de l'expression de politiques de contrôle d'accès à l'évaluation des règles à proprement parler. Le **Chapitre 7** clôture cette seconde partie par trois cas d'utilisation du prototype, représentatifs des possibilités offertes par celui-ci en terme de modularité et d'évaluation de règles multiples et variées.

Enfin, la conclusion au **Chapitre 8** est l'occasion, se basant sur les résultats obtenus par ce mémoire, de formuler les réponses aux questions de recherche initialement posées. Les possibles travaux à envisager pour approfondir cette réflexion sont finalement évoqués.

1.4 Limites du mémoire

Définissant le cadre de ce travail, nous exposons également les limites que nous nous sommes imposées ou qui se sont imposées d'elles-même dans la réalisation de ce travail. Obtenir des résultats précis dans un cadre bien défini oblige à établir que certains éléments moins essentiels ici seront ignorés :

L1 Ce travail porte uniquement sur la phase d'autorisation du contrôle d'accès, les autres aspects (notamment d'authentification ou de traçabilité) ne sont pas pris en compte.

L2 La solution logicielle construite se veut être une première esquisse et n'a pas pour vocation de respecter scrupuleusement un ensemble de conformités en terme d'architecture logicielle ou de sécurisation des ressources.

Première partie

Contexte

CHAPITRE 2

MODÈLES DE CONTRÔLE D'ACCÈS

Lorsqu'une organisation souhaite définir une politique de sécurité afin d'avoir une vue d'ensemble de la sécurié possible (et nécessaire) de ces systèmes, les **contraintes du contexte dans lequel le contrôle d'accès doit être mis en place** définissent la plupart du temps les élements cruciaux qui vont guider la méthode à suivre.

Plus spécifiquement, pour la **phase d'autorisation**, sur laquelle les efforts de ce travail se focalisent, il est nécessaire, en premier lieu, avant d'être capable de rendre une décision d'accès, d'avoir établi une politique et donc des règles de contrôle d'accès pour un système. Une règle représente des privilèges donnés à une entité (ou un groupe d'entités) pour la réalisation d'action sur une ressource (ou un groupe de ressources). C'est au moment de la définition de ces règles qu'intervient la **nécessité de recourir à un modèle** de contrôle d'accès.

L'application d'un modèle permet (à l'inverse d'une description de règles "from scratch" et sans méthode suivie) de profiter de mécanismes adaptés aux éléments du contexte du système qui ont été identifiés comme importants à prendre en compte. Ces mécanismes représentent l'avantage d'avoir déjà été éprouvés et validés. Les bénéfices de l'utilisation d'un modèle sont souvent cruciaux en vue d'exprimer une politique de sécurité de manière efficace (et notamment au niveau de la complètude de la politique et du rejet des redondances).

Deux principes généraux sont récurrents à la plupart des modèles de contrôle d'accès utilisés à l'heure actuelle :

Moindre privilège : lors de la définition de la permission d'un sujet à effectuer une action sur une ressource, il est toujours préférable d'offrir uniquement les privilèges nécessaires à l'exécution de la tâche ;

Séparation des rôles : la décomposition des actions (et des permissions associées) au juste niveau de granularité est préférable afin de permettre une découpe exacte des permissions de contrôle d'accès.

L'ensemble des modèles de contrôle d'accès qui sont décrits dans cette section respectent ces deux principes comme nous allons le voir.

2.1 Mandatory Access Control

Le Mandatory Access Control (*contrôle d'accès obligatoire* en français) s'emploie dans des contextes où la sécurité doit être très rigide (par exemple des contextes militaires).

Dans ce modèle, les droits sur les ressources sont définis à la construction du système. Chaque ressource est attachée à un **niveau de sécurité**, c'est-à-dire un niveau de classification (par exemple top secret, secret, confidentiel, non-classé) ainsi qu'une catégorie. Chaque utilisateur se voit également, sur le même mode que les ressources, attribuer un niveau de sécurité. L'utilisateur a alors accès à toutes les ressources qui ont été attachées au même niveau de sécurité que lui.

Le modèle MAC offre une grande rigueur dans la gestion des droits, ce qui est primordial dans certains systèmes, mais présente cependant certaines limites lorsqu'un utilisateur a besoin d'accéder à des ressources d'un niveau de sécurité inférieur à celui qui lui a été attribué.

Dans une version du modèle, basé sur le modèle théorique de *Bell – La Padula* [5], deux règles sont mises en oeuvre afin de définir formellement ce qui est permis :

- *No read up* : Un accès en lecture peut être accordé à un utilisateur uniquement si le niveau de sécurité de la ressource est équivalent ou inférieur à son niveau de sécurité
- *No write down* : Un accès en écriture peut être accordé à un utilisateur si le

niveau de sécurité de la ressource est équivalent ou supérieur à son niveau de sécurité

En intégrant ces deux règles, le modèle permet à un utilisateur d'accéder en lecture à des ressources de niveau inférieur mais sans pouvoir les modifier. À l'inverse, un utilisateur ne pourra accéder à une ressource de niveau de sécurité supérieur en lecture et ne pourra donc pas non plus la modifier.

Dans ce système extrèmement rigide, l'accès en écriture à des ressources de niveau inférieur n'est pas permis et demande d'outrepasser les règles du modèle. Cela est généralement possible en abaissant temporairement le niveau de sécurité d'un utilisateur.

D'autres modèle théoriques existent, sur lesquels baser un modèle de contrôle d'accès MAC. Le modèle *Biba* [6] par exemple, en définissant des règles inverses (*No read down*, *No write up*), permet de passer outre la faiblesse du modèle de *Bell – La Padula* qui ne porte que sur la confidentialité des données et ne permet pas de préserver leur intégrité dans tous les cas.

2.2 Discretionary Access Control

Le *contrôle d'accès discrétionnaire* définit, pour chaque ressource d'un système, les permissions offertes à chaque utilisateur. La notion de **propriétaire de ressources** est utilisée : l'utilisateur propriétaire décrit les droits qui sont accordés aux autres utilisateurs sur "sa" ressource.

Pour cela, on a recours à une **matrice des contrôle d'accès** (ACL), qui reprend l'ensemble des autorisations accordées à des utilisateurs sur une ressource. Chaque élément de la matrice indique une opération accordée à un utilisateur.

Par exemple, considérons trois droits sur des fichiers : lecture, écriture et exécution. Deux utilisateurs, Alice et Bob, possèdent chacun un fichier, respectivement 01 et

02. Un troisième utilisateur, Carole, ne possède pas de fichier. Alice définit que Bob peut uniquement exécuter son fichier 01 tandis que Carole possède les trois permissions. Bob permet à Alice de lire et écrire son fichier alors que Carole ne possède aucune permission dessus. Le tableau 2.I reprend les permissions accordées sur les ressources.

Tableau 2.I – matrices des contrôle d'accès

Ressource (propriétaire)	**Alice**	**Bob**	**Carole**
01 (Alice)	lecture, écriture, exécution	exécution	lecture, écriture
02 (Bob)	lecture, écriture	lecture, écriture, exécution	-

Il est également prévu, avec ce modèle, de pouvoir définir des groupes d'utilisateurs afin de regrouper des permissions équivalentes accordées sur une même ressource et de rendre l'administration des droits moins pénible.

Ce modèle est particulièrement utilisé pour la gestion des droits liés aux fichiers dans les systèmes d'exploitation.

2.3 Attribute-Based Access Control

Dans le modèle ABAC [40] (*contrôle d'accès basé sur les attributs*), les permissions d'accès sont déterminées sur base d'attributs. Pour se voir permettre l'accès à la ressource, l'utilisateur effectuant la demande doit prouver qu'un certain nombre de critères sont respectés.

Par exemple, le critère de permission d'accès peut être "avoir entre 7 et 77 ans". Lorqu'il souhaite accéder à la ressource concernée, l'utilisateur doit donc prouver qu'il est agé de plus de 7 ans et de moins de 77 ans.

Ainsi, tous les éléments qui doivent être considérés pour délivrer la décision d'au-

torisation doivent être décrits sous forme d'**attributs**. Un attribut est représenté par une paire clé-valeur (dans l'exemple précédent, pour un utilisateur de 22 ans, l'attribut pourrait être : "age"=22).

Dans le but de délivrer des autorisations d'accès, le modèle ABAC définit **trois types d'attibuts** chacun correspondant à une entité dont les caractéristiques peuvent rentrer en ligne de compte lors du processus de contrôle d'accès. Il existe donc des attributs :

- du sujet : acteur du système dont dont le nom, la fonction ou toutes autres caractéristiques (géographique, d'âge, etc) peuvent être utilisés pour rendre une décision d'accès ;
- de la ressource : composant du système (caractérisé par des attributs portant sur sa nature, son utilité ou des informations propres) sur lequel les sujets peuvent avoir une fonction à jouer ;
- de l'environnement : données liées au contexte dans lequel le contrôle d'accès est réalisé (date, heure de la journée, caractéristique de réseau ou de niveau de sécurité).

Ce modèle de contrôle d'accès présente l'énorme avantage d'être très générique et d'offrir une multitude de possiblités dans la définition des attributs qui devront être évalués. De plus, il permet une **évaluation des politiques dynamique** puisque tout élément constitutif d'un contexte ou d'une entité peut être exprimé sous forme d'attribut et donc être pris en considération dans l'évaluation.

Le langage de description de politiques XACML, qui nous détaillons au chapitre 3, est basé sur le modèle ABAC.

2.4 Role-Based Access Control

Le *contrôle d'accès basé sur les rôles* a été formalisé en 1992 par David Ferraiolo et Rick Kuhn [11].

La principale nouveauté de ce modèle est d'introduire une notion abstraite de **rôle** pour la définition des règles de contrôle d'accès. Un rôle représente le **lien entre les utilisateurs et les ressources** d'un système.

La force de ce modèle est de permettre d'intégrer directement dans les termes utilisés pour définir une politique de contrôle d'accès une sémantique propre à l'organisation visée. L'identification des rôles qui vont être utilisés dans la politique est laissée libre et offre donc la possibilité de coller au mieux au contexte de l'organisation. Ce sont souvent les compétences des utilisateurs au sein de l'organisation qui servent de base mais d'autres notions sont parfois utilisées (fonctions des utilisateurs sur un système d'information, rôles temporaires lors de la mise en place d'un projet particulier).

La définition d'une politique avec ce modèle se fait en quatre étapes :

1. définition des permissions : toutes les actions qui doivent pouvoir être réalisées sur les ressources sont listées ;

2. assignation des permissions aux rôles : les permissions précédemment définies sont associées aux rôles qui ont été choisis dans le contexte de l'organisation en vue du contrôle d'accès ;

3. assignation des utilisateurs aux rôles : les utilisateurs sont liés un à un aux rôles définis et accéderont de cette manière aux permissions qui leur incombent ;

4. désignation d'un responsable pour chaque rôle : la gestion de chaque rôle doit être réalisée par une personne compétente, qui aura la charge de veiller à ce que la définition du rôle soit correcte et que les règles d'attribution des rôles aux utilisateurs correspondent à la logique de l'organisation.

Une notion de **session** est également introduite, afin d'encadrer l'utilisation des rôles par les utilisateurs. Lors de la création d'une session, l'utilisateur **active un rôle** qui a reçu l'autorisation de réaliser l'action souhaitée sur la ressource visée. Si

un tel rôle existe et si cet utilisateur a été affecté à ce rôle, alors l'utilisateur aura la permission de réaliser cette action sur cette ressource une fois le rôle activé.

Les rôles étant définis au plus proche des fonctions des utilisateurs dans une organisation, ces notions de session et d'activation de rôles (obligeant à la **séparation statique** de ceux-ci) permettent de veiller à ce que les rôles soient utilisés sans que des incohérences naissent, notamment lorsque l'utilisation simultanée de plusieurs rôles est recquise.

Des mécanismes permettant d'organiser l'ensemble des rôles entre eux existent également. Par exemple, la création de **hiérarchies entre les rôles** permet à des rôles considérés comme au sommet de la hiérarchie d'inclure les permissions définies pour les rôles se situant plus bas dans la même hiérarchie. Il est également possible de décrire des contraintes supplémentaires (spatiales, temporelles) à prendre en compte lors de l'activation d'une session afin d'affiner la décision de contrôle d'accès.

Extensions du modèle R-BAC

Lorsque certaines contraintes fortes existent dans une organisation, le modèle R-BAC peut se révéler inefficace. Par exemple, lorsqu'il existe des différenciations géographiques entre des utilisateurs possédant un même rôle dans l'organisation, il n'est pas possible de faire la différence entre ces utilisateurs avec un modèle R-BAC "simple".

Imaginons un employé d'une agence bancaire liégeoise et un employé d'une autre succursale de la même banque à Namur. Chacun possède le rôle "gestionnaire de comptes" sur un même système centralisé. Imaginons de plus que l'on attribue à chaque agence la permission de modifier uniquement les informations des clients de sa région. Il est possible de détourner légèrement le modèle R-BAC pour définir par exemple un rôle "gestionnaire de comptes liégeois" et un autre "gestionnaire de comptes namurois". Cependant en respectant strictement le modèle R-BAC, il n'est pas possible de faire une différenciation entre les gestionnaires de comptes des

deux villes pour peu que ceux-ci soient identifiés au sein d'un système commun.

Afin de remédier à de telles limitations, il existe des extensions du modèle de base. La création de **groupes de rôles** va par exemple permettre de différencier des utilisateurs qui ne l'étaient pas avec le modèle R-BAC basique.

D'autres extensions de ce modèle existent, qui prennent comme base le modèle R-BAC afin de l'adapter et de l'agrémenter pour rencontrer les exigences propres à des domaines d'application spécifiques. Nous voyons à la section 2.5 une de ces extensions, augmentant le modèle R-BAC pour définir des politiques de confidentialité et de respect de la vie privée.

2.5 Privacy-aware Role Based Access Control

Les modèles détaillés précédemment permettent d'exprimer des politiques de contrôle d'accès et pour le modèle R-BAC, d'adapter la définition de ces politiques au contexte de l'organisation. Cependant, dans le cadre particulier de la gestion de la confidentialité des données personnelles, ces modèles ne remplissent pas les exigences nécessaires pour une expression efficace de politiques.

Une telle expression demande une finesse supplémentaire pour identifier précisément l'utilisation faite des données personnelles (par exemple récoltées pour être utilisées dans un but mais qui ne doivent pas être utilisées dans un autre sans l'accord de l'utilisateur) ainsi que pour définir des conditions et obligations propres à ce domaine.

Le modèle P-RBAC se propose de fournir un ensemble de concepts pour exprimer des **politiques de confidentialité et de gestion des données personnelles** très complexes [25]. Pour cela, le modèle introduit la notion centrale d'**attribution de permission** (*permission assignment*) afin de représenter les politiques de confidentialité. De plus, des algorithmes de détection de conflits entre attributions de

permission sont également mis en place.

Nous allons voir ici les quatre familles de concepts qui composent le modèle P-RBAC.

Le **Core P-RBAC** est le modèle de base. Il introduit les types d'entités qui seront utilisés :

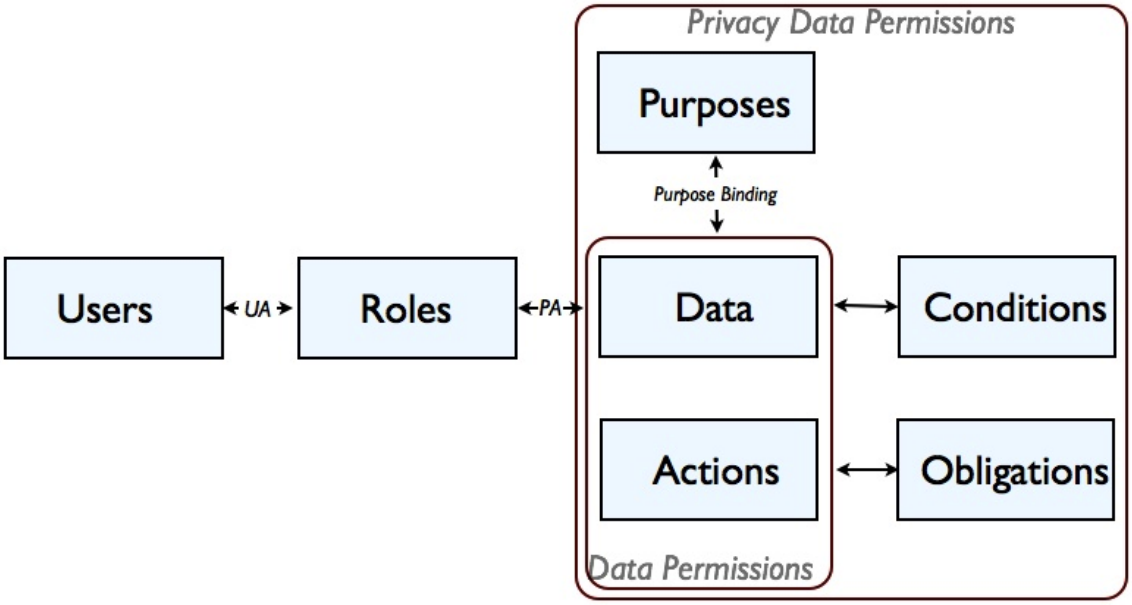
- un utilisateur : être humain ;
- un rôle : fonction ou titre dans l'organisation (comme dans le modèle RABC) ;
- une donnée : information relative à un individu ;
- une action : programme exécutable dont les invocations exécutent des fonctions pour les utilisateurs ;
- un but : terme définissant le cadre et l'objectif d'une action ;
- une obligation : action qui doit être utilisée après l'exécution d'une action sur une donnée ;
- une condition : pré-requis qui doit être rempli avant que toute action soit exécutée.

Les relations entre ces entités sont représentées à la figure 2.1.

Les conditions que l'on peut définir dans le Core P-RBAC respectent un langage à l'expressivité limitée, couvrant uniquement des domaines finis de valeurs. Des mécanismes permettant la description de conditions plus complexes sont introduits dans le Conditional P-RBAC.

Le processus à suivre pour définir une politique est le même que dans le modèle R-BAC : on définit les rôles puis on assigne les permissions aux rôles et enfin les utilisateurs aux rôles.

Figure 2.1 – schéma du Core P-RBAC



Les types d'action et de données que P-RBAC contrôle dépendent du type de système dans lequel ils sont implémentés, ce sont donc des informations de contexte qui sont prises en compte, en plus de la notion de rôle. Associés aux conditions et obligations, l'ensemble des termes offrent la possibilité de décrire des attributions de permission complexes. Ces attributions de permission du modèle P-RBAC respectent de plus les bonnes pratiques mises en place par l'OCDE [27] ainsi que les principaux textes de loi américains.

Le **Hierarchical P-RBAC** introduit des hiérarchies de rôles, de données et de buts. Il améliore de cette manière le modèle Core P-RBAC et offre des possibilités étendues sur les trois entités principales du modèle. Les hiérarchies de rôles trouvent une utilité naturelle dans la représentation des relations entre autorités et entre responsabilités au sein d'une organisation. Les hiérarchies de buts et de données paraissent tout aussi naturelles pour représenter correctement des phénomènes d'organisations.

En effet, une politique de confidentialité peut s'appliquer dans un but général alors

que des politiques (et règles) plus spécifiques devront être définies pour des sous-buts particuliers. La même logique s'applique pour les données des individus. Il peut être intéressant de définir une politique de confidentialité pour l'ensemble des données d'un utilisateur alors que certaines restrictions, différentes, s'appliqueront à des informations ou éléments du dossier.

Le **Conditional P-RBAC** introduit la notion d'ensembles d'attributions de permission et les expressions booléennes. Le but premier est de fournir un langage pour exprimer des conditions de manière plus complète qu'avec les conditions basiques qu'offre le Core P-RBAC.

Enfin, l'**Universal P-RBAC** combine les fonctionnalités des deux familles précédentes (Conditional P-RBAC et Hierarchical P-RBAC) et offre donc un modèle très complet pour décrire des politiques de confidentialité.

2.6 Organisation-Based Access Control

Les possiblités offertes par le modèle R-BAC, l'unique notion de rôle et les extensions qui ont pu être amenées pour répondre aux attentes de différents domaines, s'avèrent cependant trop limitées dans certains cas.

Par exemple, dans le cas d'un hôpital, des contraintes trop complexes existent. Les services doivent fonctionner 24 heures sur 24 heures, avec des heures de présence des médecins variables, des personnels de soin possédant également leurs horaires et ne se chevauchant pas forcément exactement avec celles des médecins, les dossiers médicaux des patients doivent parfois être accédés en urgence par des personnels qui ne sont pas forcément ceux qui ont été définis initialement, tout cela en garantissant un contrôle pour l'accès et la confidentialité des ressources.

Pour ce genre d'organisation, le *contrôle d'accès basé sur l'organisation* étend le *contrôle d'accès basé sur les rôles* en prenant **l'organisation dans son ensemble**

comme source d'information centrale. Au-delà de la notion de rôle, le modèle OrBAC propose donc une prise en compte plus forte du contexte de l'organisation en instaurant une couche abstraite complète composée de vues, de rôles et d'activités.

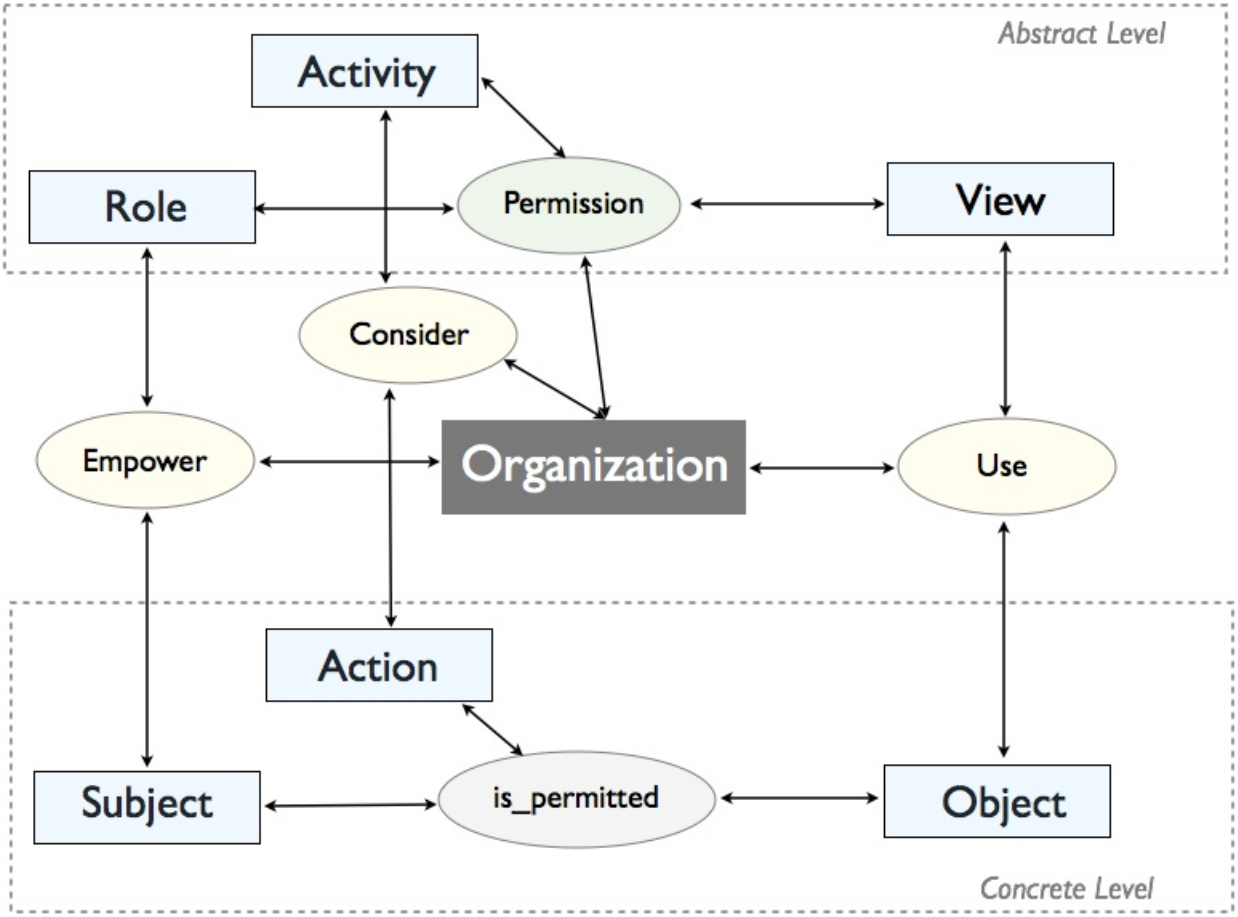
La notion de **rôle** est décrite de la même manière que dans le modèle R-BAC et est donc une abstraction de sujets (ou de leurs fonctions). La notion abstraite d'**activité** représente un ensemble d'actions qui, du point de vue des besoins en terme de sécurité, peuvent être regroupées. Enfin une **vue** regroupe de la même manière des objets qui répondent aux mêmes besoins de sécurité.

De plus, une notion de **contexte** complète ce modèle. Celui-ci peut autant être lié à une action, un sujet ou un objet. Il permet de nuancer des permissions d'accès sur base d'éléments temporels, spatiaux, etc (par exemple accéder aux informations d'un patient à l'hôpital dans une situation d'urgence).

La définition d'une politique de sécurité se fait ici complétement à deux niveaux. On définit d'abord la couche abstraite (les rôles, les activités, les vues et le contexte), qui doit représenter le plus justement la politique de sécurité établie. Ensuite on implémente cette politique en assignant des entités concrètes aux éléments de la couche abstraite. La figure 2.2 offre un aperçu de ces notions et des relations entre celles-ci.

De plus, tout comme pour les rôles dans le modèle R-BAC, il est possible de définir dans le modèle OrBAC des hiérarchies et des règles d'héritage pour les vues, les rôles et les activités.

Figure 2.2 – schéma du modèle OrBAC



CHAPITRE 3

LANGAGES D'EXPRESSION DE POLITIQUES DE SÉCURITÉ

Une fois qu'une politique de sécurité a été décrite, basée sur des critères choisis par les personnes compétentes d'une organisation et respectant un modèle adéquat, se pose la question de l'expression (efficace) de cette politique de sécurité.

Le mode d'expression choisi pour exprimer une politique sera un élément primordial en vue d'établir les modalités d'évaluation de celle-ci lors de demandes d'accès. Une politique exprimée dans un langage humain ne sera évidemment pas évaluable automatiquement et de plus, l'utilisation d'un langage non-formel augmenterait le risque de commettre une erreur lors de l'écriture et de la validation de celle-ci.

Dans le but de permettre une évaluation efficace, le recours à des *langages d'expression de politiques de sécurité* s'avère essentiel. Il existe nombre de ces langages traitables par la machine, dessinés pour des contextes très divers (expression de règles de contrôle d'accès, expression de droits numériques, sécurité des services web) et possédant différents modes de représentation (syntaxe XML, graphe RDF, langage logique).

Nous présentons ici les principaux standards des langages d'expression de politiques. Les langages présentés sont exprimables avec une syntaxe XML.

3.1 XACML

XACML (eXtensible Access Control Markup Language) est un standard du consortium OASIS (Organization for the Advancement of Structured Information Standards) qui définit une spécification en deux parties : un **langage d'expression de politiques** et un **protocole de négociation d'accès** utilisant un système de requêtes/réponses formalisées.

3.1.1 Langage de description de politiques

3.1.1.1 Modèle

Le langage d'expression de politiques XACML définit des entités très générales pour la gestion de règles de contrôle d'accès.

À la base de toute politique définie en XACML se trouve un élément **PolicySet** ou **Policy**. Un élément PolicySet peut contenir des Policy et PolicySet, ainsi que des références à des politiques potentiellement localisées sur d'autres machines. Un élément Policy représente une politique de contrôle d'accès (représentée elle-même par un ensemble de Rules, que nous définissons dans la suite). Un document XACML contient toujours un et un seul élément Policy ou PolicySet comme élément racine.

Des **algorithmes de combinaison de décisions** sont définis dans le langage afin de concilier les décisions de contrôle d'accès rendues aux vues de différentes politiques ou règles et d'en faire ressortir une unique décision unifiée. Chaque algorithme offre une manière différente de combiner les décisions. L'utilisation de ces algorithmes offre la possibilité de construire des politiques de plus en plus complexes. Au sein de la spécification XACML, il existe sept algorithmes standards et d'autres peuvent être construits au besoin.

Les principaux algorithmes de combinaison de règles sont :

- *Deny Overrides Algorithm* : si une seule évaluation de règle ou de politique renvoie la décision *Deny*, la décision de contrôle d'accès globale qui sera rendue sera obligatoirement *Deny* ;
- *Permit Overrides Algorithm* : si une seule évaluation de règle ou de politique renvoie la décision *Permit*, la décision de contrôle d'accès globale qui sera rendue sera obligatoirement *Permit* ;
- *First Applicable Algorithm* : le résultat de l'évaluation de l'ensemble des po-

litiques applicables à la requête sera le résultat de l'évaluation de la première Rule (ou Policy) applicable ;

- *Only One Applicable Algorithm* : si seulement une Policy est applicable à la requête, alors le résultat est le résultat de l'évaluation de cette Policy. Si aucune Policy n'est applicable, le résultat sera *NotApplicable* et si plus d'une Policy est applicable, le résultat sera *Indeterminate*.

Contenu dans un élément PolicySet, Policy ou Rule, un élément **Target** permet de trouver la politique qui s'applique à une requête donnée. Il est essentiellement composé d'un ensemble de conditions simplifiées relatives à un sujet (SubjectMatch), une ressource (ResourceMatch) ou une action (ActionMatch) et qui vont être appliquées aux attributs des requêtes de contrôle d'accès.

Pour cela, un élément Target va utiliser des fonctions booléennes et comparer les valeurs trouvées dans une requête avec celles que lui contient. Si toutes les conditions d'un élément Target sont remplies, alors l'élément PolicySet, Policy ou Rule qui le contient s'applique à la requête. Un élément Target peut également intégrer des informations permettant d'indexer les politiques, ce qui peut se révéler pratique si l'on souhaite stocker un grand nombre de politiques et pouvoir rapidement déterminer, lorsqu'une requête d'accès intervient, quelles politiques sont applicables.

Un élément **Rule** contient la logique de base d'une politique. Le coeur de la plupart des Rules est un (ou plusieurs) élément(s) Condition, représenté(s) par une fonction booléenne. Si la condition est évaluée à `true`, alors une décision de contrôle d'accès (une réponse contenant la décision *Permit* ou *Deny*) est retournée. Sinon, une erreur est retournée (une réponse contenant la décision *Indeterminate*) ou on signale que la condition ne s'applique pas à cette requête (une réponse contenant la décision *NotApplicable*). Une Condition peut être très complexe et construite à partir d'une imbrication arbitraire de fonctions non-booléennes et d'attributs. Par exemple, on pourrait établir une Rule qui ne permettrait un accès qu'entre neuf heures et dix-sept heures, définissant donc une intersection de deux fonctions boo-

léennes.

Un élément **Attribute** est représenté par une paire clé-valeur. La valeur doit être d'un type connu. Un attribut contient le plus souvent un identifiant de son contexte d'émission (représenté par une valeur de **AttributeId**) et une date d'émission peut aussi être précisée. Un attribut caractérise un sujet, une ressource, une action ou l'environnement propre à la requête d'accès qui est faite. Une requête d'accès est constituée quasi-exclusivement d'attributs, qui, lorsqu'elle est envoyée du PEP vers le PDP, seront comparés aux attributs des politiques (et des règles) de la base de connaissances afin de rendre une décision d'accès.

Afin qu'une Policy puisse retrouver et analyser les attributs d'une requête, il existe deux mécanismes définis dans la spécification XACML. Un **AttributDesignator** permet à une politique de désigner un attribut avec un type et une valeur donnée pour que le PDP cherche cette valeur dans la requête. Il existe quatre types de Designator, c'est-à-dire un pour chaque type d'attribut dans une requête (Subject, Resource, Action, Environment). Un **AttributeSelector** permet, lui, à une politique de chercher des valeurs particulières d'un attribut grâce à une requête XPath [31].

Des **Functions** permettent de comparer les attributs retournés par des AttributeDesignator et AttributeSelector aux valeurs attendues, afin de rendre les décisions d'accès. Les Functions peuvent être utilisées sur n'importe quelle combinaison de valeurs d'attributs et peuvent retourner n'importe quel type d'attribut reconnu par le système. Les Functions peuvent être imbriquées et opérer sur le résultat d'une autre Function. XACML définit une collection de Functions standards qui prennent un Bag (collection non-ordonnée qui accepte les duplicats, type retourné par les AttributDesignator et AttributSelector) d'un type prédéfini en entrée et retourne une valeur. C'est ces Functions que l'on retrouve dans les Conditions qui composent les règles de contrôle d'accès.

Un élément **Request** indique le sujet, la ressource, l'action (et éventuellement l'environnement) pour lesquels une requête est faite. Il y a toujours exactement une collection d'attributs Action et Resource cibles dans une requête et au plus une collection d'attributs Environment.

Enfin, un élément **Response** contient un ou plusieurs élément **Result(s)**, qui représentent chacun le résultat d'une évaluation. La grande majorité du temps, il y aura cependant un seul Result par Response. Un résultat contient la décision de contrôle d'accès, des informations de statut et optionnellement une ou plusieurs **Obligation(s)**, que le PDP a obligation de mettre en oeuvre avant d'accorder ou refuser l'accès.

Les réponses envoyées au PEP sont toujours consistantes et contiennent une des décisions d'accès suivantes :

- Permit : une (ou des) politique(s) sont applicable(s) à la requête et la décision d'accès rendue est positive ;
- Deny : une (ou des) politique(s) sont applicable(s) à la requête mais la décision d'accès rendue est négative ;
- Indeterminate : aucune décision ne peut être rendue (une erreur est survenue ou un élément d'évaluation est manquant) ;
- Not Applicable : aucune politique n'est applicable à la requête.

3.1.1.2 Profils

Les éléments du langage seuls ne suffisent pas à définir des politiques de sécurité. XACML offre des entités générales et des relations entre elles. Cependant, comme nous l'avons vu dans le Chapitre 2, il est nécessaire de recourir à un modèle de contrôle d'accès adéquat pour décrire une politique.

Nous voyons ici les principaux profils pour le langage XACML, qui permettent

d'avoir recours à des modèles de contrôle d'accès et mécanismes récurrents pour une utilisation avancée des possibilités du langage. Ces profils ont été introduits dans la version 2.0 du langage XACML. Dans les versions 1.x, seul un profil RBAC basique avait été défini.

Profil pour la description de ressources hiérarchiques

Nous envisageons ici la possibilité de traiter des ressources hiérarchiques, c'est-à-dire des ensembles de noeuds contenant des valeurs et organisés sous forme d'arbre (un noeud racine) ou sous forme de forêt (plusieurs noeuds racines), la seule exigence étant que la structure ne présente pas de cycle dans les liens entre les noeuds.

Les intérêts de pouvoir traiter ce genre de ressources sont multiples. Il peut être intéressant de pouvoir associer une même contrainte facilement à plusieurs noeuds. Dans certains cas, il peut également être nécessaire de faire dépendre l'accès à un noeud d'un autre de la hiérarchie.

Les noeuds d'une ressource hiérarchique sont traités comme des ressources individuelles. L'accès à chaque noeud est évalué et les relations entre les noeuds dans la hiérarchie (descendant, parent, feuille, noeud intérieur, etc) ne préjugent pas des décisions d'accès rendues sur chaque noeud.

Dans le traitement de ressources hiérarchiques, trois points cruciaux sont à considérer :

- représenter l'identité de chaque noeud : l'identification d'un noeud doit être faite de manière cohérente et concordante avec la représentation utilisée dans les requêtes d'accès. Pour une identification au sein d'un document XML, une *URN* (`urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-id`) spécifique à ce profil a été définie.

- effectuer une demande d'accès sur un noeud : l'accès à chaque noeud doit être

correctement défini. Si les attributs des noeuds sont mal spécifiés, l'évaluation de ceux-ci sera rendue impossible. Pour cela, un ensemble de bonnes pratiques permettent de définir de manière formelle les relations entre les noeuds.

- établir des politiques qui s'appliquent à un ou plusieurs noeuds : il existe une multitude de manière d'établir des politiques qui s'appliquent à des noeuds d'une ressource hiérarchique, pour des ressources définies en XML ou non.

Profil pour la description de ressources multiples

Il peut être souhaitable pour un PEP d'effectuer une demande unique pour l'accès à des ressources multiples ou, de la même manière, d'obtenir une réponse d'accès unique pour une requête faite sur plusieurs ressources requises.

Une telle demande pourrait être utilisée pour éviter d'envoyer plusieurs messages de demande de décision d'un PEP vers un PDP, par exemple. Alternativement, un PEP souhaitera peut-être soumettre une demande unique pour tous les nœuds d'une hiérarchie, et souhaitera alors obtenir une unique décision d'autorisation qui indique si l'accès à tous les nœuds demandés est autorisé. Un telle demande pourrait être utilisée lorsque le demandeur souhaite avoir accès à l'ensemble (ou à un sous-ensemble) des éléments d'un document XML par exemple.

Concrétement, les éléments produits par l'évaluation d'une demande d'accès à des ressources multiples doivent être identiques à ceux qui seraient produits pour une série de demandes, chaque demande d'accès contenant la référence à une des ressources. Le mappage d'une demande d'accès "composée" en des requêtes atomiques sera fait par le gestionnaire du contexte XACML. De la même manière, plusieurs décisions de contrôle d'accès seront mappées en un ensemble d'éléments Result dans la réponse.

Dans le cas d'une ressource hiérarchique, sur laquelle on soumet une requête d'accès pour tout ou partie des noeuds de la hiérarchie, on souhaite alors obtenir un seul élément Result indiquant si la requête renvoie un résultat positif pour l'ensemble

des noeuds visés par la requête. Pour cela, le processus suivi est celui d'une série de requêtes, chacune portant sur un noeud de la hiérarchie. Pour obtenir un seul résultat, représentant l'ensemble des décisions d'accès pour les noeuds de la hiérarchie, on constitue une intersection de tous les résultats, c'est-à-dire que si tous les résultats contiennent la réponse *Permit*, alors la décision finale sera *Permit*. Sinon la décision finale sera *Deny*.

Profil pour la description de politiques de confidentialité

L'OCDE définit un ensemble de bonnes pratiques [27] pour dessiner les contours d'entités préservant des informations personnelles d'utilisateurs :

- ouverture : les données personnelles obtenues doivent l'être absolument avec le consentement de l'utilisateur et dans le respect de la loi ;
- qualité des données : les données personnelles doivent être en accord avec le but prévu ainsi qu'exactes, complètes et tenues à jour ;
- définition du but : la fin pour laquelle les données personnelles sont collectées doit être précisée au plus tard au moment de la collecte des données à l'utilisateur et lors de toute modification de celle-ci ;
- limitation de l'usage : les données personnelles ne doivent pas être divulguées, ni mises à disposition dans un but différent de celui défini à l'origine ;
- garanties de sécurité : les données personnelles doivent être protégées contre des risques tels que la perte ou les accès non autorisés, la destruction, l'utilisation, la modification ou leur divulgation ;
- participation individuelle : un individu doit avoir le droit d'être informé de l'utilisation ou non de ses données, de modifier, rectifier ou effacer ses données personnelles ;
- responsabilisation : un module du système doit être responsable du respect des mesures indiquées ci-dessus.

Pour la mise en place de telles entités en utilisant la spécification XACML, deux atributs particuliers sont définis : "`urn:oasis:names:tc:xacml:2.0:resource:purpose`" et "`urn:oasis:names:tc:xacml:2.0:action:purpose`". Ils indiquent respectivement le but pour lequel une ressource a été collectée et le but pour lequel un accès à une ressource est demandé.

De plus, une **règle de contrôle d'accès standard** est définie. Elle stipule que l'accès doit être refusé à moins que le but pour lequel l'accès est demandé correspond au but pour lequel les données (qui représentent la ressource) ont été recueillies.

Profil R-BAC

Un profil pour le modèle R-BAC, que nous avons vu à la section 2.4, existe pour le langage XACML, qui détaille comment mettre en oeuvre ce modèle de contrôle d'accès avec le standard OASIS.

Pour exprimer des rôles, les éléments **Attribute** du langage XACML sont utilisés. Il revient à l'organisation définissant les rôles d'avoir recours à un identifiant d'attribut cohérent et dont la sémantique est claire. Pour cela, la spécification du profil préconise l'utilisation d'un identifiant défini sur mesure :
"`urn:oasis:names:tc:xacml:2.0:subject:role`", avec le type de données
"`http://www.w3.org/2001/XMLSchema#anyURI`".

L'assignation des rôles aux utilisateurs est faite par une **Role Enablement Authority**, lors d'une session d'utilisation. Une **Role Assignment <PolicySet>** est utilisée pour savoir quels utilisateurs ont la permission et sous quelles conditions, d'activer quels rôles. Bien qu'il n'y ait aucune obligation sur la manière de définir une Role Assignment <PolicySet>, il est recommandé d'exprimer les rôles sous forme d'attributs de ressource avec un **AttributeId** de valeur "`&role;`" et une valeur de l'attribut qui est l'URI pour le rôle donné.

Le contrôle d'accès est implémenté en utilisant deux types de politiques : **Role**

<PolicySet> et **Permission <PolicySet>**. Pour chaque rôle, une Role <PolicySet> est définie. L'élément Target de cette politique permet de rendre applicable la politique uniquement aux sujets qui possèdent l'attribut correspondant au rôle donné. Chaque Role <PolicySet> doit contenir un unique élément **<PolicySetIdReference>** qui référence la Permission <PolicySet> associée au rôle.

Dans le même temps, pour chaque rôle, une Permission <PolicySet> est définie. Elle contient les éléments Policy et Rule qui définissent les accès permis aux sujets qui se voient attribuer le rôle donné. L'élément Target de la Permission <PolicySet> ne doit pas restreindre les sujets sur lesquels elle est applicable. Cette Permission <PolicySet> ne doit jamais être utilisée par le PDP du système comme politique de sécurité initiale.

Si un rôle donné **hérite** d'autres rôles, alors sa Permission <PolicySet> associée doit contenir un élément <PolicySetIdReference> par rôle hérité qui référence la Permission <PolicySet> associée.

Une Permission <PolicySet> peut inclure un élément **HasPrivilegesOfRole <Policy>**, contenant un élément Rule avec un effet *Permit*. Si elle est mise en place, cette Rule offre la possibilité de faire des requêtes pour savoir si un sujet donné possède les permissions sur un rôle.

3.1.2 Protocole de négociation d'autorisation

Le protocole de négociation d'autorisation fourni dans la spécification XACML défini quatre modules pour constituer et gérer la base de règles et de faits, recevoir des requêtes de contrôle d'accès et rendre des décisions (sur base de réponses d'autorisation d'accès).

Classiquement, le scénario de base est qu'un utilisateur d'un système (ou d'une application) souhaite réaliser une action sur une ressource. Pour cela, il effectue une requête d'accès à cette ressource au système sur lequel il est en train d'interagir.

Au sein du protocole de négociation d'autorisation, ce système fait office de **Policy Enforcement Point**. Il reçoit la requête d'accès de l'utilisateur et forme à partir de celle-ci une requête XACML conforme (contenant au moins des informations sur l'utilisateur, la ressource et l'action à réaliser). Le PEP envoie cette requête au **Policy Decision Point**, qui a la responsabilité de rendre une décision d'accès sur base des attributs de la requête et des politiques compatibles de la base de connaissance. Une réponse contenant la décision d'accès est donc finalement envoyée en retour au PEP.

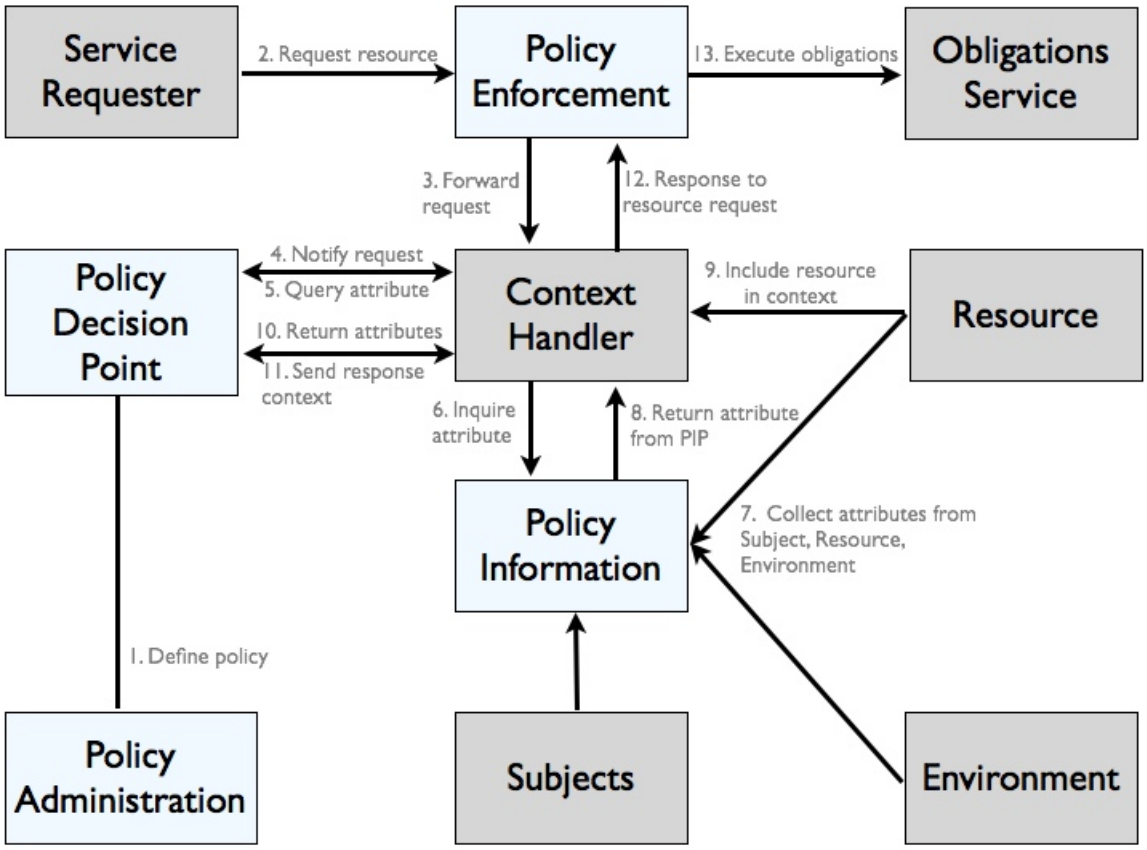
À la structure simple du protocole avec deux modules (PEP et PDP), peuvent être ajoutés deux autres modules définis également dans la spécification XACML. Le **Policy Administration Point** peut être implémenté pour gérer les politiques et règles de la base de connaissance et interagit directement avec le PDP pour lui fournir les politiques lors de l'évaluation de requêtes de contrôle d'accès. De son côté, le **Policy Information Point** collecte des informations complémentaires à la demande du PDP afin d'affiner les décisions de contrôle d'accès rendues. Les informations collectées peuvent concerner à la fois le sujet, la ressource ou l'environnement d'évaluation.

Ce protocole de négociation d'autorisation basé sur des requêtes/réponses de contrôle d'accès présente plusieurs avantages. Tout d'abord, le fait d'avoir recours à un unique composant d'évaluation de politiques permet de réduire à la fois les coûts de gestion et les risques d'incohérence (contrairement au cas où chaque application ou module possède son propre mécanisme de contrôle d'accès).

Étant un standard OASIS, la spécification est très complète et permet de résoudre nombre de problèmes qui se présenteraient avec un système moins largement diffusé. Associé au langage générique d'expression de politiques, il offre une solution complète pour la mise en place d'un contrôle d'accès centralisé.

Enfin, les différents modules du protocole peuvent être déployés ou rassemblés

Figure 3.1 – protocole de négociation d'autorisation XACML



sur une même machine selon les besoins. Nous voyons dans la suite qu'un certain nombre d'implémentations de ce protocole existent, présentant des optimisations et avantages divers.

Implémentations

- Drools expert [21] : le moteur d'inférence de règles de la suite Drools de *JBoss* est un outil puissant d'évaluation de règles métier. Il offre de très nombreuses possibilités pour la définition de règles et un moteur d'inférence à chaînage avant très puissant. Cependant, n'étant pas construit originellement pour l'évaluation particulière de règles de politiques de sécurité XACML, utiliser

cet outil demande une phase importante d'adaptation à la logique particulière du langage de politiques et de requêtes/réponses de décisions d'accès défini par la spécification XACML.

- SunXACML [30] : cette implémentation open-source de *Sun Microsystems* basée sur XACML 1.1 et écrite en Java a l'avantage d'avoir été développée sans ajout à la spécification de XACML et d'être très bien documentée. Des interfaces permettent de plus d'étendre l'implémentation avec de nouveaux attributs, fonctions et algorithmes de combinaison de politiques ;

- Xengine [7] : ce projet développé par Fei Chen (du département d'informatique de l'université du Michigan) et Alex X. Liu (du groupe de recherche en systèmes automatisés de l'université de Caroline du Nord) est basé sur la spécification XACML 2.0 et offre une implémentation du PDP. Des optimisations y sont mises en place, améliorant grandement la vitesse d'évaluation de politiques lorsque le nombre de règles présentes dans la base de connaissances augmente ;

- PicketBoxXACML [20] : le framework *Java* de sécurité PicketBox développé par *JBoss* possède un module pour la gestion des autorisations qui implémente le protocole d'autorisation XACML et offre également des performances améliorées et notamment un mécanisme de mise en cache et d'indexation de politiques pour une évaluation plus efficace. Cependant le manque de documentation rend son utilisation pas forcément évidente.

3.2 EPAL

3.2.1 Modèle

Le *Enterprise Privacy Authorization Language* d'IBM est un langage d'expression de politiques de sécurité permettant la mise en place pratique notamment de politiques de contrôle d'accès et de confidentialité. Il a été soumis, dans sa version

1.2, comme proposition de standard au consortium W3C en 2003 mais il n'a à ce jour pas encore été nommé en ce sens [4].

Dans le langage EPAL, un élément **politique** décrit des listes de hiérarchies de catégories de données, de catégories d'utilisateurs et de buts et définit des ensembles d'actions, d'obligations et de conditions.

Les principaux éléments du vocabulaire utilisé par le langage EPAL sont décrits ci-dessous :

- **catégories d'utilisateurs** : regroupent les utilisateurs qui utilisent les données ;
- **catégories de données** : regroupent les données qui sont traitées d'une même manière du point de vue de la confidentialité ;
- **buts** : modélisent les services pour lesquels les données ont été collectées ;
- **actions** : définissent comment les données sont utilisées ;
- **obligations** : représentent des actions qui doivent être mises en oeuvre par le système EPAL ;
- **conditions** : sont des expressions booléennes qui permettent d'évaluer les éléments du contexte ;
- **règles** : sont des assertions mentionnant une décision de contrôle d'accès sur base d'une catégorie d'utilisateurs, d'une action, d'une catégorie de données et d'un but. Une règle peut également contenir des conditions et obligations.

Une fois constituée et suivant les éléments qui viennent d'être décrits, une politique EPAL est principalement composée d'un ensemble de règles de sécurité ordonnées et hiérarchisées.

EPAL pouvant s'appliquer à l'ensemble des domaines de l'entreprise qui nécessitent une prise en compte de la confidentialité, aucune sémantique n'est prédéfinie pour

les valeurs à donner aux différents éléments du modèle du langage. Cependant, un mécanisme est prévu pour aider à la définition de termes.

Tout comme le standard OASIS XACML, les entités du langage EPAL sont exprimables dans une syntaxe XML. Un schéma XML décrivant l'ensemble de ces entités est d'ailleurs disponible [1].

Enfin, le **processus d'évaluation de politiques** sur base de requêtes d'accès sur lequel se fonde la spécification EPAL est assez similaire au modèle utilisé par la spécification XACML. En effet, ces deux modèles respectent des standards communs provenant de la même norme ISO [12].

Les notions de *Policy Enforcement Point* et de *Policy Decision Point*, que l'on a déjà évoqué dans la description de XACML font partie des éléments de base de cette norme et sont également repris dans le processus d'évaluation utilisé par EPAL.

3.2.2 Limites

En analysant le fonctionnement et les entités offertes par EPAL, il ressort que celui-ci offre des possibilités de description de politiques beaucoup moins étendues que le standard XACML. Il ne permet ainsi d'exprimer qu'un petit sous-ensemble des fonctionnalités offertes par XACML.

De plus, dans sa version 2.0, le langage XACML 2.0 fournit également un profil pour l'expression de politiques de confidentialité (détaillé à la section 3.1.1.2). De ce fait, l'ensemble des politiques qui peuvent être définies en EPAL peuvent l'être avec des éléments XACML.

Enfin, XACML étant reconnu comme standard, il bénéficie des révisions d'experts d'organisations diverses et présente en cela une assurance supplémentaire.

[1] `http://www.zurich.ibm.com/security/enterprise-privacy/epal/Specification/index.html`

CHAPITRE 4

LANGAGES D'EXPRESSION DE DROITS

Les langages d'expression de droits (REL) constituent un sous-ensemble des langages d'expression de politiques de sécurité. Ils sont destinés à la gestion des droits relatifs à des biens numériques (DRM). Ces langages permettent de définir des licences pour tout type de biens numériques (morceau de musique, livre électronique, image, etc), sous forme de permissions détenues sur ces biens. L'ajout de contraintes de coût, de durée ou d'autres types sur les permissions proposées est le plus souvent supporté.

Permettant d'exprimer dans des représentations évaluables automatiquement des droits liées à des ressources numériques, ces langages présentent un intérêt important pour toutes sortes de distributions de contenus en ligne.

4.1 ODRL

Le *langage ouvert d'expression de droits numériques* est une initiative, lancée dans les années 2000 par Renato Iannella, pour la définition d'un langage ouvert d'expression de droits numériques.

Bien qu'il ait été développé dans ce cadre à ses débuts, le langage a depuis évolué et poursuit maintenant le but plus large de devenir un **standard ouvert pour l'expression de droits** pour l'ensemble des acteurs de la communauté DRM. De plus, une communauté W3C pour ODRL a été créée pour guider l'initiative [1].

4.1.1 Version 1.1

Dans sa première version [18], le langage est centré sur trois entités principales :

[1] `http://www.w3.org/community/odrl`

- les **rights** : sont composés des permissions, contraintes, informations sur l'ayant droit, informations sur le bien distribué, contraintes et toutes autres informations qui décrivent la licence exprimée ;
- les **asset** : représente un contenu physique ou digital. Le plus important est que le bien puisse être correctement identifié ;
- les **party** : peut représenter à la fois les ayants droit (un individu, une organisation ou tout autre type d'entité) qui proposent un bien ou les utilisateurs finaux qui souhaitent obtenir des permissions sur le bien.

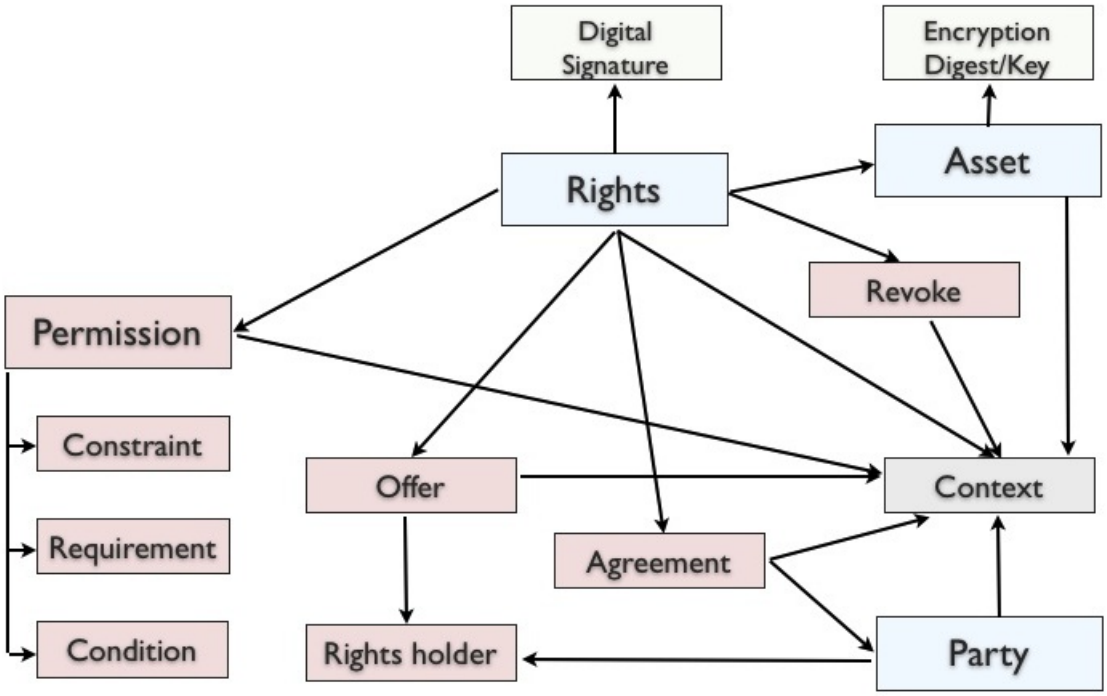
Avec ces trois entités principales, la langage ODRL permet d'exprimer des licences dans deux buts différents. Les **offer** sont les propositions des ayants droit pour obtenir des droits sur un bien. Les **agreement** représentent la réalisation d'une offre, acceptée par une partie (un utilisateur) pour un bien donné. En complément, le modèle du langage permet d'exprimer une notion de révocation de n'importe quelle offre ou accord.

De plus, toutes les entités principales du modèle offrent la possiblité d'exprimer des informations supplémentaires (sur les entités ou des relations entre elles) en utilisant un élément **context**. L'utilisation de cet élément peut être judicieux dans de nombreux cas, dépendant de chaque type d'entités. Utilisé sur l'élément racine d'une licence (l'élément *rights*), il peut par exemple permettre d'identifier de manière unique l'entièreté d'une expression. C'est également avec cet élément que les autres types d'entités peuvent être identifiés.

Enfin, les **constraint** permettent de représenter des restrictions sur les permissions qui sont offertes (ou accordées) sur les biens. Il existe plusieurs catégories de contraintes déjà définies pour indiquer des restrictions très diverses (sur les utilisateurs susceptibles d'acquérir des permissions sur un bien, sur les usages associés à une permission, des contraintes temporelles, dans les nombres d'usages permis du bien, etc).

La structure offerte par l'ensemble des éléments du modèle, représentée à la figure 4.1 est très complète et flexible. Elle permet de supporter la description de licences pour tout type de biens numériques dans des contextes de distribution et de consommation très variés.

Figure 4.1 – schéma du modèle ODRL 1.1



Ce modèle est exprimé sous la forme de deux schémas XML : un schéma principal [2] qui décrit toutes les entités du modèle et les relations entre elles et un schéma "dictionnaire" [3] qui décrit des termes qui peuvent être utilisés dans des cas concrets. Ces schémas (et surtout le schéma "dictionnaire") peuvent être étendus afin de définir de nouveau termes. C'est de cette manière que des profils d'utilisation vont être décrits pour le langage.

Plusieurs profils très complets ont été définis pour la version 1.1 de ODRL et offrent

[2] `http://odrl.net/1.1/ODRL-EX-11.xsd`
[3] `http://odrl.net/1.1/ODRL-DD-11.xsd`

donc des cas précis d'utilisation du langage où une manière d'utiliser le modèle est spécifiée et des nouveaux termes de vocabulaire décrits.

OMA DRM REL

Le premier profil que nous présentons a été mis en place par l'Open Mobile Alliance [4] dans le cadre du développement du système de gestion de droits numériques de la même organisation : OMA DRM.

Ce profil [2] se base sur les éléments du modèle ODRL pour spécifier une sémantique de droits favorisant l'utilisation sur appareils mobiles de contenus protégés par des DRMs. Étant défini pour une utilisation de licences dans des environnements mobiles, ce profil privilégie des licences simples et vise la définition de droits quelque soit le type de contenu ou les méchanismes de transports sollicités.

Concrétement, la spécification du profil consiste principalement à lier aux contenus sur lesquels des droits sont exercés des informations de protection ainsi qu'à expliquer l'usage particulier qui doit être fait des droits et des contraintes ODRL dans ce contexte.

Ainsi, seules les permissions `display`, `print`, `play` et `execute` du dictionnaire du langage de base ODRL sont conservées. Une permission `mode` et deux contraintes `system` et `mode` sont également définies.

Creative Commons Licenses Profile

Un second profil intéressant défini sur ODRL 1.1 est le profil pour la définition de licences Creative Commons [19] avec ODRL.

Les licences Creative Commons sont constituées par combinaison de caractéristiques qui ont été prédéfinies pour représenter l'ensemble des cas de distribution et

[4] `http://www.openmobilealliance.org`

de partage de contenus. Ces caractéristiques donnent une sémantique pour établir des permissions, prohibitions et exigences liés à l'utilisation de ces contenus.

Le langage ODRL définit également des entités pour des permissions et des exigences. De ce fait, celles-ci peuvent être utilisées pour **décrire les permissions et exigences définies dans les licences Creative Commons**. De plus, les prohibitions Creative Commons peuvent être exprimées comme des contraintes ODRL.

Afin d'établir un lien entre les expressions définies en ODRL et évaluables automatiquement et les versions textuelles des licences, il est possible d'indiquer un **lien vers la version textuelle** dans l'élément `context` de l'expression ODRL d'une licence. Il faut cependant veiller à ce que l'expression ODRL et le texte de la licence référencée correspondent exactement. Si des contraintes non définies dans la licence Creative Commons ont été ajoutées à une expression ODRL, alors celle-ci devient plus spécifique que la licence Creative Commons et ne doit pas en référencer le lien.

De plus, la définition de contraintes sur les biens, droits ou parties d'une licence à l'aide du nouveau vocabulaire décrit par le profil doit se faire en veillant à ne pas rentrer en **conflit avec des termes du modèle ODRL**. En particulier, une utilisation inadaptée du profil pourrait amener à des chevauchements de sémantiques entre termes du vocabulaire du profil et des termes du modèle ODRL. Il n'y a dans ce cas aucune solution technique qui prévale et l'attention du rédacteur est donc essentielle.

4.1.2 Version 2.0

Dans sa seconde version [16], ODRL est toujours exprimé d'une part, sous forme d'un ensemble d'entités qui forment la structure du langage et d'autre part, grâce à des termes définis dans un dictionnaire de vocabulaire commun (définissant des actions, contraintes et devoirs).

Cependant, le modèle du langage évolue et on ne parle plus de *rights* mais de **policy**. Cette entité principale du modèle est de plus ici exprimée en termes de permission et de prohibition. La figure 4.2 représente ces entités ainsi que les autres éléments du modèle avec les relations entre eux.

Une **permission** permet l'exécution d'une action sur un bien donné. Une **prohibition**, symétriquement, interdit l'exécution d'une action. La deuxième différence avec une permission est qu'aucun devoir ne peut être défini pour une prohibiton. Au-delà, les deux entités fonctionnent de la même manière.

Une **party** peut représenter à la fois l'ayant droit distribuant son bien et l'utilisateur acquérant des droits sur le bien. Le mécanisme utilisé ici pour marquer la différence entre les deux utilisations de l'entité est la définition d'un attribut **function** pour chaque partie. Un ayant droit sera désigné `assigner` alors qu'un utilisateur final sera lui désigné `assignee`.

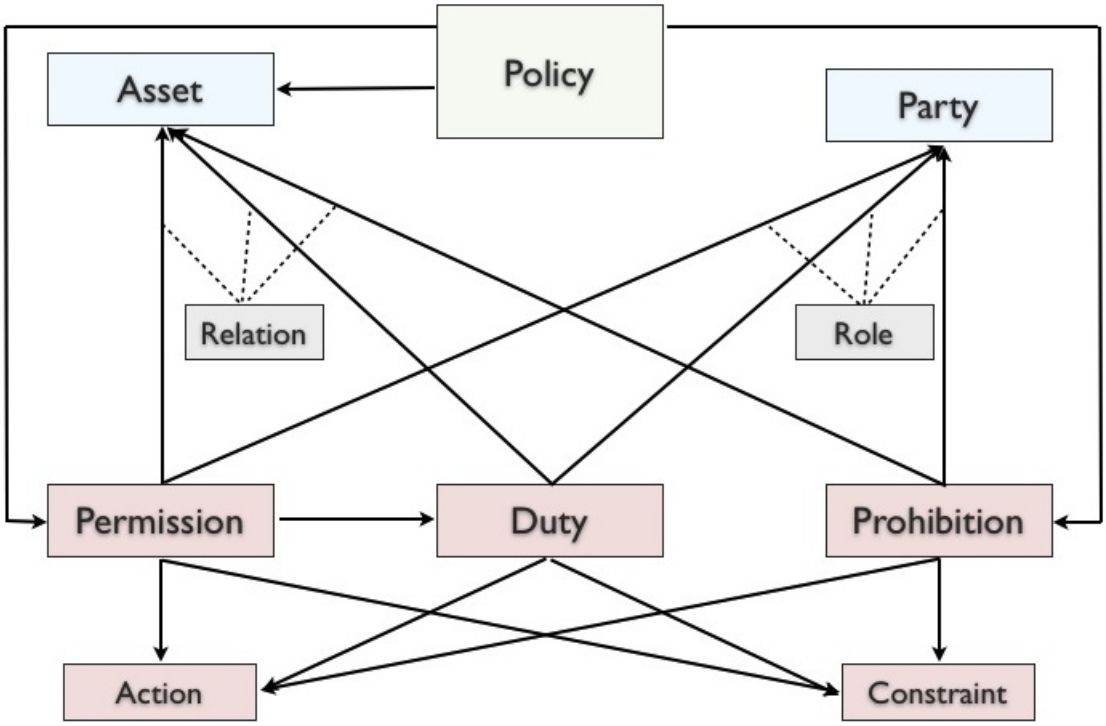
Un **asset** identifie un contenu sur lequel les permissions et prohibitions qui sont exprimées dans la politique s'appliquent. Un bien peut également être lié à un devoir duquel il est alors partie prenante de la définition.

Dans une permission, une **action** indique l'opération que la partie désignée `assignee` possède l'autorisation de réaliser sur le bien associé. Dans une prohibition, l'action indique évidemment ce qui est interdit à la partie désignée `assignee` de réaliser. Dans la définition d'un *duty*, l'action indique l'opération qui doit être réalisée.

Une **constraint** décrit une limite ou restriction qui doit obligatoirement être considérée. Elle peut être définie pour une permission, une prohibition ou pour un devoir à respecter. Une contrainte est composée (au moins) d'un nom, de deux opérandes et d'un opérateur mathématique. De plus, si plusieurs contraintes sont associées à une entité, alors toutes ces contraintes doivent être satisfaites lorsque la permission, la prohibiton ou le devoir donné(e) sont pris en considération.

Un **duty** représente une exigence qui doit être menée à bien lorsque l'on obtient une permission sur un bien. Il indique une action à opérer et une partie peut être spécifiée lorsque le rôle que l'on considère n'est pas évident (dans le cas de rôles multiples). Un bien peut également être référencé, qui est alors un des éléments de la définition du devoir, par exemple dans le cas où il est nécessaire de stipuler un montant à payer.

Figure 4.2 – schéma du modèle ODRL 2.0



La spécification finale de ODRL 2.0 étant finale depuis seulement quelques semaines, peu de profils existent actuellement qui utilisent ce langage et soient déjà à un stade suffisamment avancé pour être étudiés et utilisés concrétement.

RightsML profile

Un des profils les plus avancés au jour d'aujourd'hui est celui qui est développé par l'International Press Telecommunications Council [5] [8] pour la distribution de contenus de presse par les agences de presse, publicateurs, intermédiaires et clients sur les marchés numériques.

Pour la définition de ce profil, certaines recommandations sont apportées quant à l'utilisation des entités du modèle ODRL 2.0 :

- l'utilisation de *policy* de type `set` est préférable ;
- la valeur de l'attribut `uid` de l'entité *asset* peut représenter soit une ressource, soit une catégorie de ressources ;
- la valeur de l'attribut `uid` de l'entité *party* peut représenter soit une partie, soit une catégorie de parties.

Certains termes définis dans le dictionnaire commun de ODRL 2.0 sont de plus exclus car ils se révèlent inappropriés dans le contexte d'utilisation visé par le profil. Le fait d'exclure ces termes du profil indique principalement que s'ils sont utilisés pour spécifier des licences ODRL, celles-ci ne pourront être considérées comme conformes au profil et la responsabilité de leur utilisation correcte reviendra aux personnes qui ont pris la liberté de les utiliser.

La liste complète des termes exclus du profil RightsML est reprise dans la spécification de celui-ci. Cela concerne principalement des valeurs d'actions et quelques valeurs de contraintes.

Enfin, des actions et attributs de contraintes sont également ajoutés par le profil RightsML au vocabulaire recommandé.

Creative Commons Licenses profile

Comme cela était le cas pour la version 1.1 de ODRL, l'expression de licences Creative Commons est une possibilité intéressante à définir dans un profil pour

[5] `http://www.iptc.org`

ODRL 2.0. Aucun profil n'étant encore défini pour cette version, une tentative de définition de celui-ci a été faite en parallèle de ce mémoire, dans le but de créer un cas simple et concret d'utilisation de ODRL 2.0.

L'ensemble des spécifications de ce profil sont reprises dans le document présenté à l'annexe I de ce mémoire.

4.2 MPEG-21 REL

Le langage d'expression de droits MPEG-21 REL fournit une méthode pour spécifier des droits associés à la distribution et l'usage de biens (contenus, services, données, etc).

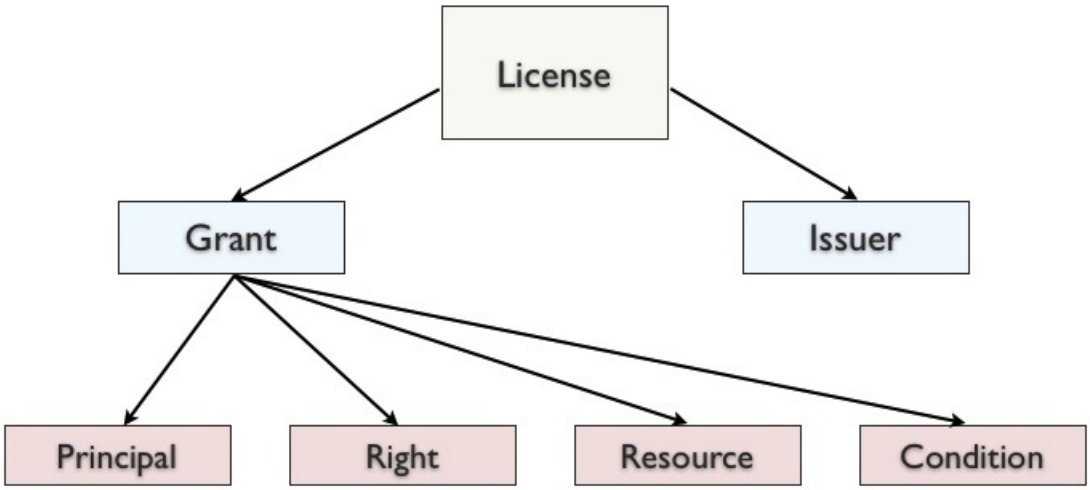
Il est à la base d'un standard international ISO développé par le Moving Picture Expert Group[6] (MPEG). Il a été défini en se basant sur des éléments provenant du langage XrML dans sa version 2 [36] mais a depuis été étendu et affiné. De plus, les brevets sous-jacents aux notions du langage appartiennent au groupe ContentGuard [7], qui regroupe des acteurs importants (Microsoft, Time Warner, Thomson) et dont le but est de contribuer à la création de standards dans le domaine des DRMs.

Comme cela est représenté à la figure 4.3, l'entité principale du langage est l'élément *license*. Une licence est la représentation de droits proposés sous forme de *grants* par un distributeur.

Un **issuer** unique est spécifié pour une licence. Il doit être correctement identifié et un certain nombre d'informations complémentaires peuvent être décrites, comme une signature électronique permettant de faciliter l'établissement de relations de confiance entre distributeurs et utilisateurs finaux.

[6] `http://mpeg.chiariglione.org`
[7] `http://www.contentguard.com`

Figure 4.3 – schéma du modèle MPEG-21 REL



Un **grant** spécifie qu'une partie principale a un droit sur une ressource, potentiellement sous certaines conditions. Un élément *license* peut contenir plusieurs *grants*, qui seront toujours associés au même distributeur.

La **principal** peut représenter un utilisateur, un groupe d'utilisateurs, un appareil ou n'importe quel système qui exerce le droit sur la ressource. Des informations pour référencer un mécanisme d'authentification (par exemple une clé privée provenant d'une paire clé publique/clé privée) peuvent être ajoutées sous forme d'un élément *keyHolder* pour prouver formellement l'identité d'une partie.

Le **right** associé à un *grant* représente l'action qui peut être exercée sur la ressource. Un certain nombre de droits sont définis dans le dictionnaire du langage.

La **resource** est l'objet sur lequel des droits sont cédés et peut être autant un contenu numérique que par exemple un service ou une information textuelle (une adresse email, un rôle ou une propriété).

Des **conditions** peuvent être définis sur les *grants*. Il est possible de décrire des conditions simples (un intervalle de temps pour l'exercice d'un droit, le nombre de

fois qu'un média peut être utilisé) ou des conditions plus complexes basés sur des droits prérequis. Enfin, l'intersection de plusieurs conditions est également considéré comme une condition.

La norme ISO spécifiant MPEG-21 définit également la spécification d'un **dictionnaire de valeurs** pour le langage d'expression de droits, avec une méthodologie pour la définition de nouveaux termes à utiliser dans des contextes particuliers. La définition du dictionnaire est étroitement liée à la spécification du modèle, afin d'assurer une utilisation conforme à celui-ci dans la définition de nouveaux termes pour la caractérisation d'autres types de contenu. Le dictionnaire amène aux propriétaires de contenu une vraie facilité pour utiliser directement le langage.

Enfin, deux extensions du langage ont déjà été décrites pour apporter des notions complémentaires de définition de droits. Au delà du modèle central du langage, l'**extension standard** offre un ensemble de concepts généraux applicables dans de nombreux cas d'utilisation de DRM. L'**extension multimédia** fournit pour sa part des termes spécifiques aux domaines des contenus multimédia comme les ebooks, morceaux de musique numérique ou vidéos.

REL Mobile profile

Comme tous les standards ISO, la spécification du langage MPEG-21 REL n'est pas disponible librement. Deux profils sont définis au sein de celle-ci mais ne possédant pas d'accès à cette spécification, nous préférons ici présenter un profil simplifié mais dont les détails sont disponibles librement. Nous présentons donc le REL Mobile Profile, qui est défini par ContentGuard comme un **exemple de profilage** de leur langage [35].

La cheminement de la création de ce profil suit l'approche générale décrite pour la création de profils pour MPEG-21 REL :

1. commencer avec l'entiéreté des éléments du langage (modèle et extensions standards) ;

2. sélectionner, élément par élément, ceux qu'il est cohérent de garder pour une utilisation dans le domaine d'application visé ;

3. restreindre, lorsque cela est nécessaire pour correspondre aux exigences du domaine, les cardinalités des éléments gardés ;

4. restreindre, lorsque cela est nécessaire pour correspondre aux exigences du domaine, les valeurs permises pour chaque entité ;

5. ajouter, lorsque cela est nécessaire pour correspondre aux exigences du domaine, les valeurs nécessaires au nouveau profil.

Ce profil permet de montrer que les concepts MPEG-21 REL sont adaptés pour répondre aux exigences particulières de calcul limité sur des appareils mobiles.

Concrètement, un sous-ensemble des éléments du modèle de base est conservé et certaines cardinalités sont restreintes. Ceux-ci sont repris dans le tableau 4.I.

Tableau 4.I – éléments du profil Mobile pour MPEG-21 REL

Élément	**Cardinalités dans le profil**	**Cardinalités dans le modèle**
license		
grant	1..non borné	0..non borné
issuer	1..1	0..non borné
profileCompliance	0..1	0..1
grant		
keyHolder	0..1	0..1
play/print/execute	1..1	1..1
diReference	1..1	0..1
validityInterval		
validityIntervalFloating		
exerciseLimit		
allConditions	0..1	0..1
keyHolder		
info	1..1	1..1
diReference		
identifier	0..1	1..1
allConditions		
condition	0..non borné	0..non borné
validityInterval		
notBefore	0..1	0..1
notAfter	0..1	0..1
validityIntervalFloating		
duration	1..1	0..1
exerciseLimit		
count	1..1	0..1
issuer		
keyHolder	1..1	0..1

Deux contraintes supplémentaires sont de plus ajoutées pour compléter les exigences du profil. Celles-ci sont reprises dans le tableau 4.II.

À l'aide de tous ces éléments, des licences peuvent être exprimées en utilisant le langage MPEG-21 REL tout en respectant les contraintes particulières de la diffusion de contenus sur des appareils mobiles.

Tableau 4.II – contraintes supplémentaires pour le profil Mobile pour MPEG-21 REL

Élément	**contrainte additionnelle**
license	Un élément *license* peut contenir plusieurs *grant* mais toutes doivent être liées au même utilisateur final
keyHolder	Son élément-fils `info` ne doit contenir qu'un et un seul élément KeyName

4.3 Interopérabilité entre RELs

Comme cela a été évoqué dans l'introduction de ce travail, au-delà de la description de chaque langage et de l'utilisation propre qui peut en être faite, il est intéressant, afin de créer des possibilités de communication et d'échange entre différents environnements, d'analyser l'interopérabilité possible entre langages.

Nous présentons dans cette section les différentes tentatives qui ont déjà été menées pour l'interopérabilité entre les principaux langages d'expression de droits ainsi qu'une solution particulière qui a été envisagée en se basant sur le langage d'expression de politiques XACML.

4.3.1 Entre le profil DAC MPEG-21 REL & OMA REL

Une première manière d'envisager l'interopérabilité entre RELs est de prendre comme point de départ les profils définis sur ceux-ci. En effet, pour certains domaines d'application, différents langages offrent des solutions d'expression de licences et se trouvent donc d'une certaine manière concurrents pour délivrer une spécification efficace des droits.

Dans un but d'interopérabilité, il est alors intéressant d'analyser les similitudes et divergences entre les solutions proposées et d'en tirer des **possibilités d'équivalence entre les sémantiques** propres.

Entre les langages MPEG-21 REL et ODRL, plusieurs analyses ont déjà été faites concernant des profils des deux langages qui pourraient concorder. Par exemple,

le profil DAC de MPEG-21 REL permet de représenter des droits d'utilisation de programmes de télévision diffusés numériquement. De son côté, le sous-ensemble OMA DRM du langage ODRL présente un profil spécifique pour la gestion des droits de contenus diffusés numériquement [23].

Plus précisement, la version 2 de OMA DRM définit la notion de **Broadcast Rights Objects** (BCROs), qui sont des éléments destinés à être diffusés à des récepteurs (des utilisateurs finaux) qui ne possèdent pas de mécanisme de réponse. Ces BCROs sont donc diffusés de manière répétée afin que les récepteurs concernés augmentent leur chance de recevoir les objets. Ces objets sont utilisés afin de fournir une protection à des contenus sous forme de DRM, dans le but de garantir une diffusion efficace de ceux-ci.

De leur côté, les mécanismes décrits par le profil DAC de MPEG-21 REL sont généralement utilisés par des services de diffusion pour communiquer avec des récepteurs matérialisés par des Digital TeleVision (DTV). Dans ce cadre, le profil spécifie tout une série de droits et de conditions (contrôle du signal, contrôle d'accès simultanés, spécification du niveau de sécurité) utilisables conjointement avec des contenus à diffuser protégés par DRM.

L'analyse des deux profils montre que les types de permissions utilisés par les BCROs dans le profil Broadcast pour OMA DRM peuvent trouver des équivalents parmi les éléments du profil DAC pour MPEG-21 REL. Ces correspondances sont reprises dans le tableau 4.III.

Tableau 4.III – correspondances permissions profil Broadcast pour OMA DRM / profil DAC pour MPEG-21 REL

profil Broadcast pour OMA DRM	**profil DAC pour MPEG-21 REL**
o-dd :play	mx :play
o-dd :display	mx :play
o-dd :execute	mx :execute
o-dd :print	mx :print
<oma-dd :export mode="move"/> <oma-dd :export mode="copy"/>	<m1x :governedMove/> <m2x :governed-Copy/>
oma-dd :access	mx :play
oma-dd :save	m1x :governedCopy

Pour les contraintes utilisées dans le profil Broadcast pour OMA DRM, nombre de correspondances directes sont également possible. Seules certaines fonctions de OMA DRM qui ne font pas partie des exigences du profil DAC de MPEG-21 REL ne sont pas supportées. Les correspondances réalisées sont reprises dans le tableau 4.IV.

Tableau 4.IV – correspondances contraintes profil Broadcast pour OMA DRM / profil DAC pour MPEG-21 REL

profil Broadcast pour OMA DRM	**profil DAC pour MPEG-21 REL**
<o-dd :count> COUNT </o-dd :count>	<sx :exerciseLimit> <sx :count> COUNT </sx :count> </sx :exerciseLimit>
<oma-dd :timed-count timer="SECOND"> COUNT </oma-dd :timed-count>	<m2x :timedExerciseLimit> <m2x :duration> PERIOD </m2x :duration> <m2x :count> COUNT </m2x :count> </m2x :timedExerciseLimit>
<o-dd :datetime> <o-dd :start> </o-dd :start> <o-dd :end> </o-dd :end>	<r :validityInterval> <r :notBefore> START_DATE_TIME </r :notBefore> <r :notAfter> END_DATE_TIME </r :notAfter> </r :validityInterval>
<o-dd :interval> PERIOD </o-dd :interval>	<sx :validityIntervalFloating> <sx :duration> PERIOD </sx :duration> </sx :validityIntervalFloating>
<o-dd :individual> <o-ex :context> <o-dd :uid> id </o-dd :uid> </o-ex :context> </o-dd :individual>	<m1x :identityHolder> <m1x :idValue> id </m1x :idValue> </m1x :identityHolder>

Cet exemple de tentative d'interopérabilité entre les profils de deux langages pour un même domaine d'application montre que des équivalences sont souvent possibles et qu'ainsi des échanges et la mise en place de mécanismes de mappages d'un langage à l'autre peuvent être envisagés.

Cependant, une telle démarche basée sur des profils d'un domaine particulier **limite forcément les résultats** si l'on vise une interopérabilité entre langages au-delà d'un domaine précis.

4.3.2 Entre MPEG-21 REL et ODRL

Une seconde approche pour l'interopérabilité entre MPEG-21 REL et ODRL 1.1 peut être envisagée en considérant les éléments des langages directement, sans prendre en compte un profil particulier et donc l'interprétation des entités de leurs modèles pour un domaine particulier.

Tout d'abord, les deux langages sont fondés sur des bases relativement proches. La première chose commune aux deux langages est d'être basés sur une syntaxe XML (comme tous les autres langages que nous analysons dans ce travail).

De plus, même si les termes employés par ODRL semblent davantage à même de correspondre naturellement aux termes utilisés dans la réalité, la **structure d'une licence** est composée dans les deux cas de quatres éléments principaux :

- un **sujet** : l'acteur qui réalise certaines opérations. Représenté par l'élément *party* en ODRL et *principal* en MPEG-21 REL
- un **droit** : l'opération qui peut être menée sur l'objet. Représenté par l'élément *permission* en ODRL et *right* en MPEG-21 REL
- un **objet** : le contenu pour lequel des droits sont décrits. Représenté par l'élément *asset* en ODRL et *resource* en MPEG-21 REL

- des **conditions** : précisent dans quel contexte le droit peut être exercé. Représenté par l'élément *constraint* en ODRL et *condition* en MPEG-21 REL

Ces similitudes semblent indiquer qu'une traduction pourrait être envisagée entre les deux langages. En effet, même pour des licences exprimées avec un grand nombre de termes (contraintes pour l'exercice des droits proposés, utilisateurs multiples, etc), on peut imaginer que la structure générale des deux langages étant très proche, les difficultés du mappage d'un langage à l'autre résideront surtout dans la réalisation de correspondances exactes entre les sémantiques proposées.

Concrétement, afin de travailler à un mappage au niveau des licences exprimées, les technologies XML existantes et particulièrement XSL [34] peuvent se révéler adaptées.

Comme cela est réalisé pour des licences simples et dans les deux sens [22], la traduction se fait alors au niveau syntaxique en indiquant les *patterns* à repérer dans le langage source et en créant la structure d'une licence dans le langage cible pour y placer les données (signature du distributeur, nom de l'objet considéré, valeurs primitives utilisées par les contraintes, etc).

Cette seconde approche pour l'interopérabilité entre ODRL et MPEG-21 REL offre des perspectives plus larges en proposant de traduire l'ensemble des éléments d'un langage dans un autre. La méthode utilisée se révèle plus axée sur la syntaxe que sur la sémantique des éléments dans un contexte donné, comme l'indique l'utilisation de la technologie XSL pour décrire les règles de mappage.

Cependant, cette interopérabilité est pour l'instant effective surtout pour des licences simples, l'élargissement de la méthode étant envisagé. De plus, le mappage des éléments de ODRL avec ceux de MPEG-21 REL ne permet pas d'envisager une structure plus large pour l'interopérabilité au-delà de ces deux seuls langages.

4.3.3 Interopérabilité basée sur XACML

La dernière tentative pour l'interopérabilité entre RELs que nous abordons ici se veut plus originale. En effet, le *Distributed Multimedia Applications Group (DMAG)* [8], un groupe de recherche de l'université polytechnique de Catalogne qui travaille sur la mise en place de solutions pour la gestion et la distribution de contenus multimédia de manière sécurisée et interopérable, présente une solution basée sur un **système central d'évaluation en XACML** [37].

Le langage XACML est ici considéré comme un sur-ensemble de tous les RELs. L'hypothèse est faite de parvenir à traduire l'ensemble des éléments des langages ODRL 2.0 et MPEG-21 REL en des termes du modèle XACML. Le principe est qu'une fois ces **correspondances établies et formalisées**, le système accepte alors des requêtes exprimées dans les termes des RELs d'origine. Ces requêtes sont traduites en XACML puis transmises au système d'évaluation en XACML. L'évaluation se fait sur les nouvelles règles des RELs d'origine traduites en XACML, et en utilisant donc un système d'évaluation entiérement et uniquement fondé sur XACML.

Afin de rendre cette solution effective, l'étape la plus importante est de parvenir à réaliser le mappage entre l'ensemble des éléments des RELs visés et le langage XACML.

L'étude qui décrit cette tentative de mise en place d'un coeur d'évaluation en XACML pour l'interopérabilité entre RELs se base sur une version intermédiaire de ODRL, qui est proche de la version 2 finale mais diffère tout de même par certains éléments. La **traduction des éléments *rights* et *permission* de ODRL en XACML** est triviale alors que les *party*, *action* et *asset* demandent d'avoir recours aux mécanismes courants de XACML pour retrouver des attributs de requêtes d'accès, à savoir les *Match* et *AttributeDesignator*. La traduction de l'élément ODRL *constraint* peut, selon le type de contraintes à exprimer, demander l'utili-

[8] `http://dmag.ac.upc.edu`

sation également d'*AttributeDesignator*.

Les tableaux 4.V, 4.VI, 4.VII, 4.VIII et 4.IX reprennent l'ensemble des traductions proposées.

Tableau 4.V – correspondances rights et permission ODRL en XACML

élément ODRL	**en ODRL**	**en XACML**
rights	<o-ex :rights [...]> [...] </o-ex :rights>	<xacml :Policy> [...] </xacml :Policy>
permission	<o-ex :permission [...]> [...] </o-ex :permission>	<xacml :Rule> [...] </xacml :Rule>

Tableau 4.VI – correspondance party ODRL en XACML

party	<o-ex :party> <o-ex :context> <o-dd :uid> subjectId </o-dd :uid> </o-ex :context > </o-ex :party>	<xacml :Subject> <xacml :SubjectMatch MatchId ="urn :oasis :names :tc :xacml :1.0 :function :string-equal"> <xacml :AttributeValue DataType="string"> SubjectId </xacml :AttributeValue> <xacml :SubjectAttributeDesignator AttributeId="..." DataType="string"/> </xacml :SubjectMatch> </xacml :Subject> </xacml :Subjects>

Tableau 4.VII – correspondance action ODRL en XACML

action	<o-dd :play/>	<xacml :Policy> <xacml :Actions> <xacml :Action> <xacml :ActionMatch MatchId ="urn :oasis :names :tc :xacml :1.0 :function :string-equal"> <xacml :AttributeValue DataType="string"> play </xacml :AttributeValue> <xacml :ActionAttributeDesignator DataType="string" AttributeId="urn :oasis :names :tc :xacml :1.0 :resource :xpath"/> </xacml :ActionMatch> </xacml :Action> </xacml :Actions> </xacml :Policy>

Tableau 4.VIII – correspondance asset ODRL en XACML

asset	<o-ex :asset> <o-ex :context> <o-dd :uid> resourceId </o-dd :uid> </o-ex :context> </o-ex :asset>	<xacml :Resources> <xacml :Resource> <xacml :ResourceMatch MatchId =”urn :oasis :names :tc :xacml :1.0 :function :string-equal”> <xacml :AttributeValue DataType=”integer”> resourceId </xacml :AttributeValue> <xacml :ResourceAttributeDesignator DataType=”string” AttributeId=”urn :oasis :names :tc :xacml :1.0 :resource :xpath”/> </xacml :ResourceMatch> </xacml :Resource> </xacml :Resources>

Tableau 4.IX – correspondance constraint ODRL en XACML

constraint	<o-ex :constraint> <o-dd :spatial o-ex :type = ”prism :vocabs/ISO3166/ES”> </o-ex :constraint>	<xacml :Condition> <xacml :Apply FunctionId =”urn :oasis :names :tc :xacml :1.0 :function :string-equal”> <xacml :AttributeSelector DataType=”string” RequestContextPath = ”//xacml-context :Resource/xacml- context :ResourceContent/location/country”/> <xacml :AttributeValue DataType=”string”> ES </xacml :AttributeValue> </xacml :Apply> </xacml :Condition>

Les éléments du langage **MPEG-21 REL** sont, de la même manière, traduits vers XACML en utilisant les mécanismes courants du langage. Il est d'ailleurs à noter que, comme pour ODRL, ce sont les entités représentant les licences et les droits qui possèdent la traduction la plus aisée. Ces détails de traduction sont repris dans les tableaux 4.X, 4.XI, 4.XII, 4.XIII, 4.XIV et 4.XV.

Tableau 4.X – correspondance licence MPEG-21 REL en XACML

élément MPEG-21 REL	**en MPEG-21 REL**	**en XACML**
license	<r :license licenseId=”1”> ... </r :license>	<xacml :Policy PolicyId=”urn :oasis :names :tc :xacml :2.0 :ex :policyid :1”> ... </xacml :Policy>

Tableau 4.XI – correspondance grant MPEG-21 REL en XACML

grant	<r :grant> [...] </r :grant>	<xacml :Rule> [...] </xacml :Rule>

Tableau 4.XII – correspondance principal MPEG-21 REL en XACML

principal	<r :keyHolder> <r :info> <dsig :KeyName> subjectId </dsig :KeyName> </r :info> </r :keyHolder>	<xacml :Subjects> <xacml :Subject> <xacml :SubjectMatch MatchId="[...]string-equal"> <xacml :AttributeValue DataType="string"> subjectId </xacml :AttributeValue> <xacml :SubjectAttributeDesignator AttributeId="subject-id" DataType="string"/> </xacml :SubjectMatch> </xacml :Subject> </xacml :Subjects>

Tableau 4.XIII – correspondance action MPEG-21 REL en XACML

action	<mx :play/>	<xacml :Actions> <xacml :Action> <xacml :ActionMatch MatchId="string-equal"> <xacml :AttributeValue DataType="string"> play </xacml :AttributeValue> <xacml :ActionAttributeDesignator DataType="string" AttributeId="xpath"/> </xacml :ActionMatch> </xacml :Action> </xacml :Actions>

Tableau 4.XIV – correspondance resource MPEG-21 REL en XACML

resource	<mx :diReference> <mx :identifier> resourceId </mx :identifier> </mx :diReference>	<xacml :Resources> <xacml :Resource> <xacml :ResourceMatch MatchId="string-equal"> <xacml :AttributeValue DataType="integer"> resourceId </xacml :AttributeValue> <xacml :ResourceAttributeDesignator DataType="string" AttributeId="xpath"/> </xacml :ResourceMatch> </xacml :Resource> </xacml :Resources>

Tableau 4.XV – correspondance condition MPEG-21 REL en XACML

condition	<r :allConditions> <sx :territory> <sx :location> <sx :country xmlns :iso="country"> Country </sx :country> </sx :location> </sx :territory> </r :allConditions>	<xacml :Condition> <xacml :Apply FunctionId="and"> <xacml :Apply FunctionId="string-equal"> <xacml :Apply FunctionId="[...]string-equal"> <xacml :AttributeSelector DataType="string" RequestContextPath="country"/> </xacml :Apply> <xacml :AttributeValue DataType="string"> Country </xacml :AttributeValue> </xacml :Apply> </xacml :Condition>

Finalement, utiliser le langage XACML qui, comme nous l'avons vu à la section 3.1, offre une grande expressivité en tant qu'"intermédiaire" d'évaluation, semble permettre une **réelle avancée pour l'interopérabilité** complète entre ODRL et MPEG-21 REL. Cette solution laisse de plus la porte ouverte à l'interopérabilité avec d'autres langages dont les éléments seraient mappables en XACML.

Deuxième partie

Prototype pour un moteur d'évaluation de règles

CHAPITRE 5

CONCEPTION

La cadre de travail étant mis en place, des choix peuvent commencer à être faits à partir de l'ensemble des technologies qui ont été décrites et analysées afin de dessiner les contours d'un prototype de moteur d'évaluation de règles qui réponde aux exigences de généricité et d'efficacité décrites plus tôt.

Dans ce premier chapitre de la seconde partie, nous commençons donc par détailler les raisonnements qui ont été suivis et qui ont menés aux choix de conception du moteur d'évaluation. Les principales questions portent sur le module central du prototype, à savoir le **module d'évaluation de politiques**. Des différentes technologies présentées dans la première partie du travail peuvent ressortir tout un ensemble de solutions, répondant différemment aux priorités définies. Nous suivrons évidemment celles qui nous paraissent les plus à même de supporter une évaluation du plus grand nombre de règles de manière efficiente.

De plus, donner au moteur d'évaluation la capacité d'évaluer le plus de types de règles possible demande également d'affiner une ligne de conduite qui permette de rester dans un **cadre simple**, favorisant l'accessibilité de la solution. L'approche qui est proposée présente cette qualité tout en laissant la porte ouverte à des améliorations futures.

Au niveau de l'implémentation, nous n'hésitons pas à tirer partie de projets et de technologies déjà existants, dans le but, d'une part, de ne pas redéfinir inutilement des choses qui ont déjà été faites et d'autre part, de profiter des facilités offertes par des outils adaptés et parfaitement reconnus.

Enfin, les **composants du prototype** dont la modélisation est guidée par les éléments définis dans les sections qui la précèdent sont présentés.

5.1 Choix pour la conception logique

5.1.1 Évaluation des politiques

Parmi les technologies présentées dans la première partie de ce travail, un certain nombre permettent de mettre en place un système complet de gestion et d'évaluation de politiques.

Il serait tout d'abord possible de considérer chacun des **langages d'expression de droits** que nous avons détaillé au chapitre 4 et de concevoir un module par langage, ayant pour but d'évaluer les règles exprimées à l'aide de celui-ci. Dans un souci d'extensibilité du moteur d'évaluation, cette solution paraît évidemment très peu adaptée, obligeant à ajouter des modules pour chaque nouveau langage supporté et rendant ainsi le composant d'évaluation toujours plus fourni.

Reprenant les résultats des tentatives d'interopérabilité entre RELs décrits à la section 4.3, il pourrait être imaginable d'améliorer ce concept. Lorsque des correspondances strictes sont possibles entre l'ensemble des éléments de deux langages dans le cadre défini d'un profil (c'est-à-dire qu'une interopérabilité totale et sans perte d'information est possible entre les profils de deux langages), il serait possible d'établir des règles de transformation des éléments d'un langage dans l'autre. De cette manière, la possibilité pourrait être offerte d'évaluer toutes les règles relevant du même domaine d'application dans un seul langage, diminuant potentiellement le nombre de modules différents d'évaluation nécessaires.

Cependant, aux vues des résultats des tentatives d'évaluation décrites aux sections 4.3.1 et 4.3.2, cette solution souffre de limitations dues à l'expressivité trop faible des RELs analysés et à la complexité d'établir des correspondances entre l'ensemble de leurs éléments et dans l'ensemble des cas d'utilisation possibles. De plus, la gestion d'un système basé sur un coeur d'évaluation composé de multiples modules pourrait vite se révéler très complexe.

L'utilisation du langage **EPAL** et du processus d'autorisation d'accès qui est défini dans sa spécification est une seconde possibilité. Le processus d'autorisation d'accès utilisé est conforme à un standard ISO [4] et les entités du modèle du langage fournissent des possibilités intéressantes de définition de politiques. Cependant, ce langage est principalement orienté pour la définition de règles de confidentialité et, par sa structure, ne présente pas une extensibilité conséquente. Il ne semble donc pas être le meilleur candidat.

Le **standard XACML**, présenté à la section 3.1, possède lui l'énorme avantage d'être basé sur un modèle de contrôle d'accès très ouvert, ABAC [40]. Par cette caractéristique ainsi que sa structure à la fois complète et générique, il offre donc des possibilités d'expressivité très élevées. Comme nous l'avons déjà souligné, il est de plus reconnu comme standard et bénéficie des retours d'information d'une très large communauté d'experts.

De plus, au fur et à mesure de ses années d'existence, de nombreux profils ont été définis pour répondre aux attentes particulières de différents types d'organisation. Un des principaux est le profil défini pour mettre en oeuvre le modèle de contrôle d'accès R-BAC, utilisé dans beaucoup d'organisation. Cependant, des profils plus spécifiques ou basés sur des modèles de contrôle d'accès un peu moins répandus mais tout aussi nécessaires existent [1] [9] [39].

Enfin, comme cela a été décrit à la section 4.3.3, la tentative d'interopérabilité entre ODRL et MPEG-21 REL menée par le groupe de recherche *DMAG* [37] et qui se base sur une évaluation en XACML associée à des "traducteurs" de règles, semble être celle qui offre le plus d'assurance de généricité et d'extensibilité.

Une dernière possibilité à envisager est l'utilisation de la technologie **Drools Expert** [21]. En effet, ce moteur d'inférence de règles métier, décrit comme implémentation possible du *Policy Decision Point* XACML à la section 3.1.2, peut être considéré comme un moteur d'évaluation "générique" pour lequel on décrirait sous

forme de règles déclaratives l'ensemble des contraintes imposées par un langage particulier.

Cette solution paraît intéressante mais n'a pas été envisagée en détail ici. En effet, la possibilité d'utiliser XACML paraît plus simple à mettre en place et permet de ne pas s'aventurer dans l'utilisation d'une technologie externe au monde des langages d'expression de politiques et dont les nécessités d'adaptation paraissent, au premier abord, assez lourdes.

Le tableau 5.I résume les principaux éléments de réflexion concernant les différentes solutions analysées.

Tableau 5.I – résumé des solutions pour l'évaluation des politiques

technologie	**avantages**	**inconvénients**
multiples modules d'évaluation	- évaluation possible dans le langage d'origine	- interopérabilité entre RELs incomplète - extensibilité limitée - multiples modules d'évaluation
EPAL	/	- manque d'expressivité du langage - langage axé politiques de confidentialité
XACML	- expressivité du langage - module unique d'évaluation	/
Drools Expert	- possibilités offertes par les règles déclaratives -moteur d'inférence puissant	- traduction des contraintes de chaque langage en règles métier très fastidieux - technologie indépendante du monde des politiques de contrôle d'accès

La solution choisie ici est donc d'utiliser pour notre module d'évaluation le langage XACML. Se basant sur des premiers résultats déjà obtenus pour l'interopérabilité entre RELs basée sur ce langage, nous formulons l'hypothèse que l'ex-

pressivité du langage permet d'**exprimer sans perte d'information un grand nombre de langages d'expression de droits et de politiques** et qu'il est donc envisageable de constituer un coeur d'évaluation unique pour l'évaluation de multiples types de règles exprimées dans des langages divers.

Les éléments démontrant cette hypothèse seront fournis dans la suite du document, notamment par la description de cas concrets au chapitre 7.

5.1.2 Support de nouveaux types de règles

Le second aspect important à considérer pour la mise en place du prototype est l'approche à adopter pour l'ajout du support de nouveaux types de règles.

En effet, le but premier du prototype est d'être le plus générique possible, à savoir de pouvoir réaliser l'évaluation de règles étant exprimées dans un maximum de langages et selon un maximum de profils de domaines d'application. Il existe donc **deux dimensions à prendre en compte** : le langage d'expression et le profil relatif au contexte de définition des politiques.

La manière de considérer ces deux dimensions au moment de dessiner les contours du prototype aura des répercussions au niveau de chaque fonctionnalité du système. Plusieurs manières de faire peuvent être envisagées :

- **considérer distinctement le support de langages et le support de profils** :
  - Le support d'un nouveau langage serait d'abord ajouté au prototype, par l'importation d'une description formelle de l'ensemble des entités du modèle du langage.
  - Un langage supporté par le prototype mais sur lequel aucun profil ne serait défini ne permettrait pas l'évaluation de règles, aucun domaine d'application particulier n'étant spécifié.

  - Lorsque l'on souhaiterait avoir la possibilité d'évaluer un nouveau type de règles (un profil défini sur un domaine d'application particulier) exprimé grâce à ce langage, on ajouterait au prototype une description du sous-ensemble des éléments du modèle et des contraintes additionnelles qui doivent être considérées.
  - L'ajout du support d'un nouveau type de règles se ferait donc en deux temps.

- **considérer le support de profils de langages** :
  - L'ajout du support d'un nouveau type de règles se ferait en une seule phase.
  - Un profil est décrit en utilisant un sous-ensemble des entités du modèle d'un langage et potentiellement en définissant des éléments et contraintes additionnels.
  - Le support d'un nouveau profil de langage se ferait en important la description formelle des éléments utilisés du langage qui supporte le profil et des potentiels éléments et contraintes additionnels.
  - Ainsi, l'ensemble des entités d'un langage ne serait jamais considéré qu'au travers de leur utilisation dans les descriptions des différents profils importés.

Quelque soit la solution retenue, il faudra, à l'évaluation, que le système puisse connaître (automatiquement ou qu'on lui indique) à la fois le langage et le profil considérés. De plus, des règles pour un même domaine d'application (profil) pourront potentiellement être décrites dans plusieurs langages.

Dans un souci de simplicité, nous choisissons ici de considérer le **support de profils de langages en une seule phase**. En effet, cette solution permet l'importation d'un seul tenant des éléments d'un profil et simplifie à la fois la structure de la base de connaissances et la désignation des éléments nécessaires à l'évaluation.

En résumé, la flexibilité offerte par cette solution rencontre au mieux les qualités principalement visées du prototype (généricité et extensibilité).

5.2 Choix pour la conception physique

5.2.1 Implémentation du moteur d'évaluation

Affinant petit à petit les différents points essentiels de la conception du prototype de moteur d'évaluation de règles, il est nécessaire de définir également sur quels éléments d'implémentation (existants ou à développer) nous pouvons baser cette conception.

Pour le moteur d'évaluation de règles XACML, il est possible d'implémenter soi-même le module de prise de décision (PDP) ainsi que les modules de gestion des politiques (PEP, PAP, PIP). Cependant, plusieurs implémentations existent, et notamment pour le PDP, qui ont été citées lors de la présentation du langage XACML (à la section 3.1.2).

Ne nous focalisant pas ici sur les performances du moteur d'évaluation ou sur d'autres optimisations annexes disponibles, notre choix s'est porté sur la solution la plus simple et qui est aussi l'implémentation la plus diffusée, à savoir celle de Sun Microsystems : **Sun's XACML**.

5.2.2 Format de représentation des profils

Pour l'ajout de profils au prototype, il est nécessaire de définir les formats de représentation acceptés et les principes sur lequels se baseront les mappages de politiques vers XACML.

Ainsi, les schémas des profils devront être exprimés en utilisant des descriptions ***XML Schema Definition*** [33] (*Document Type Definition* [32] pourrait être un format acceptable également mais nous avons réduit les possibilités à un type de

schéma par simplicité).

Afin de décrire les transformations des schémas des profils en des entités XACML, plusieurs solutions sont également envisageables. Un langage de programmation comme Java pourrait être utilisé pour réaliser cette tâche [17]. Cependant, nous préférons une fois de plus avoir recours à des technologies XML, et en particulier ***Extensible Stylesheet Language*** [34] [13], profitant des fonctionnalités simples et adaptées qu'offre cette technologie.

Par ce choix de ne recourir qu'à des technologies XML pour définir les profils et leurs patrons de transformation vers XACML à intégrer au prototype, **tout langage de contrôle d'accès ou de description de politiques de sécurité exprimable en XML**, sous réserve que l'ensemble de son modèle soit transformable en entités XACML valides, est donc candidat potentiel à l'intégration au système.

5.2.3 Composants du prototype

Les différents composants du système ont été définis sur base des décisions prises pour la conception logique puis pour l'implémentation du système. Un aperçu de ces composants est donné à la figure 5.1, sous forme d'un diagramme de composants UML [29].

Le prototype est séparé en cinq composants principaux : `Importer`, `Validator`, `Storage`, `Transformer` et `Policy Decision Point`.

Le composant `Importer` est le point de départ pour l'ajout du support de nouveaux profils ou de politiques. C'est le composant qui permet de désigner un fichier contenant le schéma XSD d'un profil (et le schéma XSL de transformation associé dans le cas d'un profil défini sur un autre langage que XACML) ou un fichier XML décrivant une politique. Il contient un sous-composant propre à chaque type de fichier à importer et communique avec le `Validator` afin de lui transmettre un (ou des) fichier(s) pour vérification de leur conformité.

Le composant `Validator` se charge de valider la conformité avec la syntaxe XML de tous les types de fichiers qui sont importés dans le prototype, ce qui est le rôle de son premier sous-composant. Un second sous-composant a pour but, dans un second temps, lors de l'importation de fichiers contenant des politiques, de vérifier la conformité de ceux-ci avec le schéma du profil sur lequel est basée l'expression de la politique (schéma disponible auprès du composant `Storage`). Dans cette première version du prototype, le profil associé à une politique est simplement indiqué par l'utilisateur lors de l'importation de celle-ci. Si les fichiers à importer sont correctement validés, ils sont alors transmis au composant `Storage`.

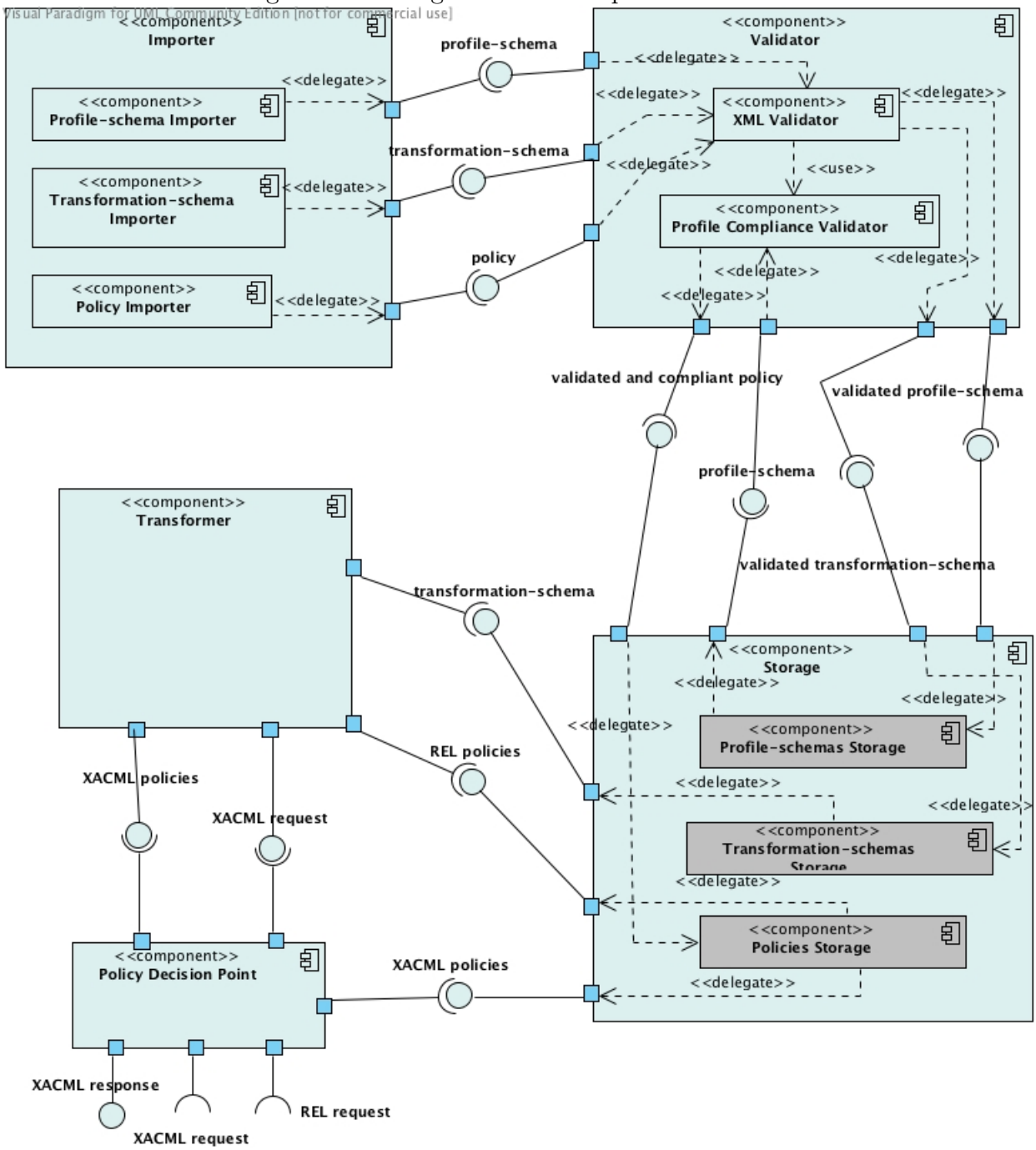
Le composant `Storage` fait office à la fois de base de connaissances du système (stockant l'ensemble des politiques déjà importées dans le système) et d'entité de stockage pour les fichiers (schémas et feuilles de transformation) décrivant les profils reconnus par le système et utilisés pour des évaluations de politiques. Le prototype ne possédant pas, dans cette version, de module de vérification de l'évaluabilité des profils et politiques importés, les données importées dans le système doivent être, de ce point de vue, vérifiées par l'utilisateur lui-même.

Le composant `Transformer` est à la **base de la stratégie d'évaluation générique et extensible définie** pour le prototype. Il est activé à la demande du `Policy Decision Point`, pour la transformation en XACML de requêtes et de politiques décrites dans d'autres langages. Pour cela, ce même composant est dévolu à la fois à la transformation de requêtes et à la transformation de politiques en XACML. Pour les deux fonctionnalités, la feuille de transformation du profil concerné par l'évaluation est obtenue du composant `Storage`. Pour la transformation de politiques de sécurité, sont également obtenues du composant `Storage` les politiques de la base de connaissances relatives au profil dont une requête est évaluée. La validité des requêtes d'accès en terme de syntaxe XML et selon profil utilisé est vérifiée à l'aide du composant `Validator`.

Le composant `Policy Decision Point` sert à l'évaluation des politiques sur base

des requêtes fournies par les utilisateurs souhaitant obtenir des autorisations d'accès. Le coeur de son implémentation est directement tiré du PDP fourni par le projet *SunXACML*. Pour ce prototype, l'utilisateur doit indiquer lui-même au système quel profil sera visé par l'évaluation. La décision de contrôle d'accès rendue à l'utilisateur porte sur l'ensemble des règles applicables à la requête donnée (toutes les règles des politiques de la base de connaissance définies sur le profil indiqué).

Figure 5.1 – diagramme de composants



CHAPITRE 6

PRÉSENTATION DU SYSTÈME

L'ensemble des modules du système tels qu'ils ont été définis au chapitre précédent permettent d'intégrer à la base de connaissances et d'évaluer sur base de requêtes de contrôle d'accès des règles conformes à des profils de langages.

Nous présentons ici, pour chaque fonctionnalité du système, la manière dont les modules du prototype de moteur d'évaluation développé permettent de la mettre en oeuvre. Pour cela, les processus de ces différentes fonctionnalités tels qu'elles sont traitées par le prototype sont modélisés à l'aide de diagrammes d'activités UML [29].

Une méthode de description de schémas pour l'intégration de profils conforme aux exigences du prototype est de plus détaillée.

6.1 Description d'un nouveau profil

Bien que cela ne fasse pas partie des fonctionalités offertes par le prototype en lui-même, nous définissons ici la méthode à suivre pour constituer des schémas décrivant des profils qui soient conformes pour une intégration au prototype. Suivre ces étapes pour exprimer un nouveau profil est donc un **prérequis** pour l'utilisation du prototype.

Nous nous appuyons sur l'approche générale pour la création de profils fournie en complément de la spécification du langage MPEG-21 REL [35]. Cette approche avait déjà été évoquée à la section 4.2 de ce mémoire et est reprise ici, généralisée à n'importe quel langage :

1. débuter avec l'entièreté des éléments du langage choisi ;

2. sélectionner, élément par élément, ceux qu'il est cohérent de garder pour une utilisation dans le domaine d'application visé ;

3. restreindre, lorsque cela est nécessaire pour correspondre aux exigences du domaine, les cardinalités des éléments gardés ;

4. restreindre, lorsque cela est nécessaire pour correspondre aux exigences du domaine, les valeurs permises pour chaque entité ;

5. ajouter, lorsque cela est nécessaire pour correspondre aux exigences du domaine, les valeurs nécessaires au nouveau profil.

Une fois ces étapes appliquées, doit alors être constitué un **schéma XSD** contenant l'ensemble des éléments subsistants du modèle du langage et provenants des définitions additionnelles spécifiques au profil. De cette manière, nous obtenons une description unifiée du profil.

De plus, pour des profils définis sur un langage qui n'est pas XACML, il est nécessaire de constituer un **schéma de transformation au format XSL** contenant les patrons de transformation pour l'ensemble des éléments du profil vers des éléments XACML valides.

La réalisation de ce schéma étant inhérente à chaque profil, aucune méthode n'est donnée pour cette étape.

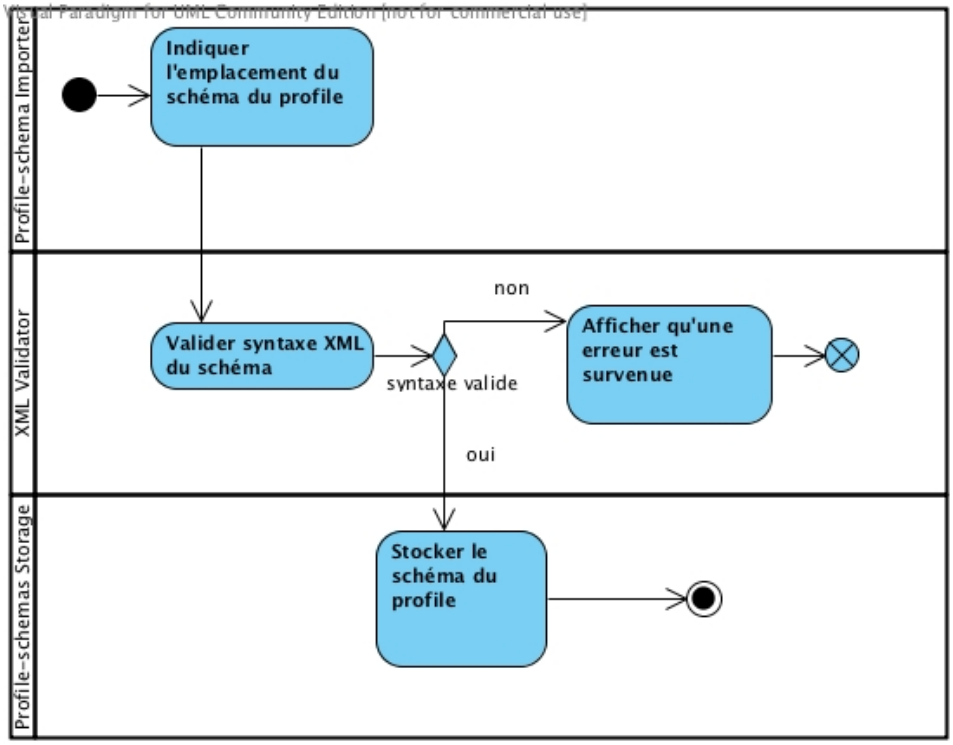
6.2 Support d'un nouveau profil

Lorsque l'on souhaite ajouter des politiques à la base de connaissance, il est impératif qu'un profil pour le type de politique choisi pour représenter le domaine concerné ait précédemment été intégré au prototype.

6.2.1 Profil XACML

Le diagramme d'activités correspondant à l'ajout du support d'un nouveau profil XACML au prototype est disponible à la figure 6.1.

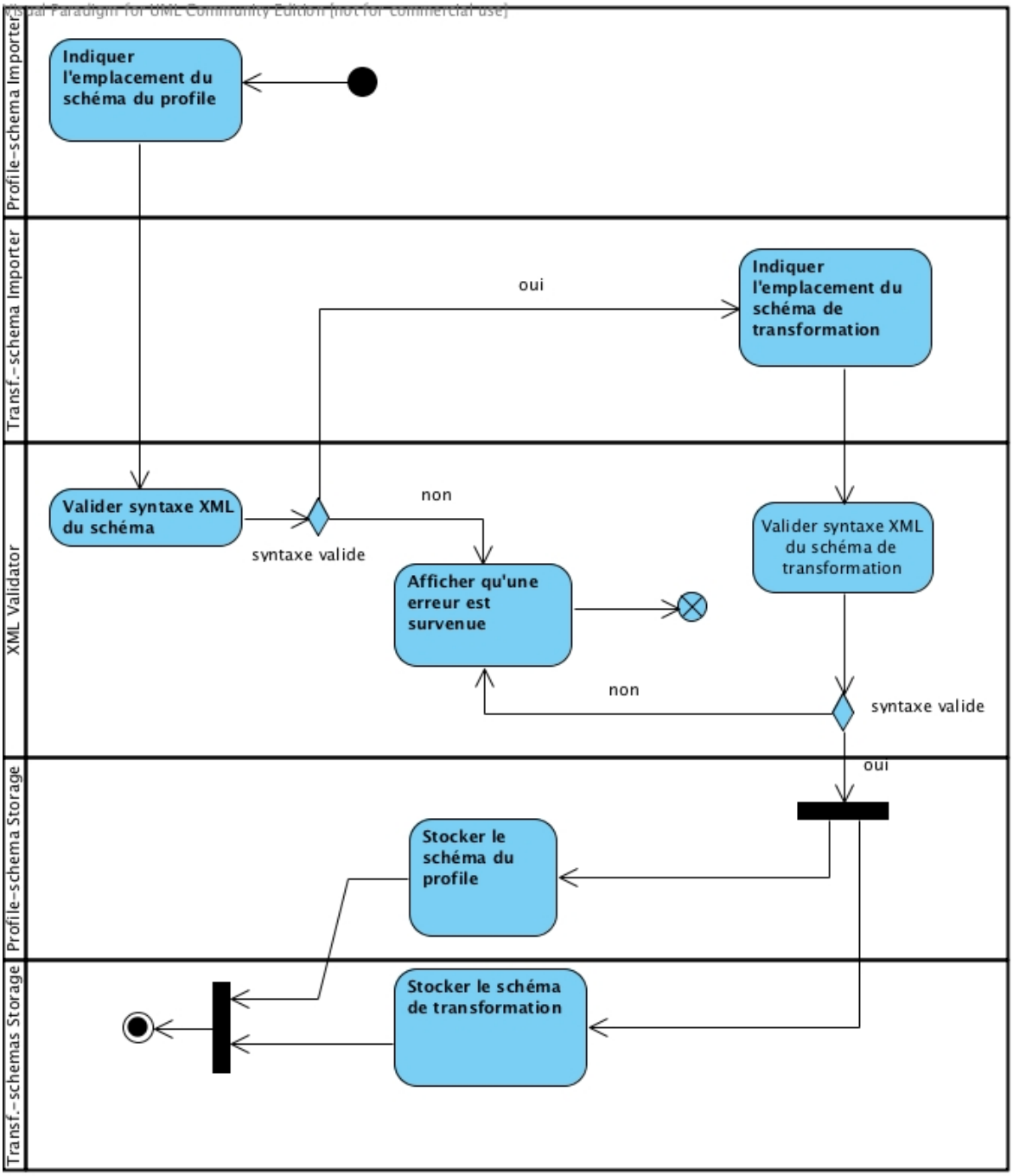
Figure 6.1 – ajout du support d'un nouveau profil XACML



6.2.2 Profil basé sur un REL

Le diagramme d'activités correspondant à l'ajout du support d'un nouveau profil basé sur un REL au prototype est disponible à la figure 6.2.

Figure 6.2 – ajout du support d'un nouveau profil basé sur un REL

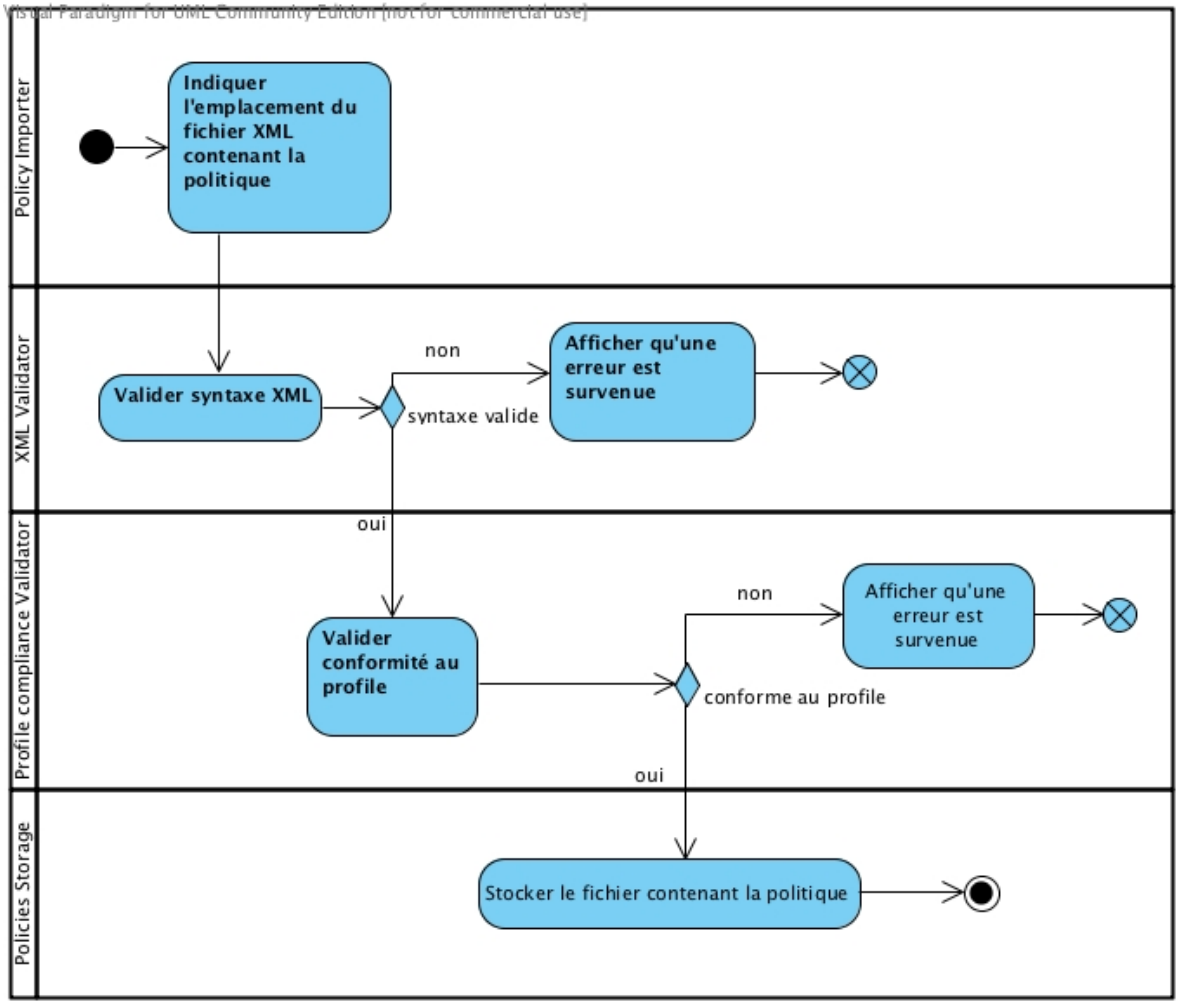


6.3 Ajout de politique à la base de connaissances

L'ajout d'une politique à la base de connaissances se fait de manière à ce que sa conformité avec la syntaxe XML et avec la structure du profil choisi soient vérifiées avant toute chose.

Le diagramme d'activités correspondant à l'ajout d'une politique à la base de connaissances du prototype est disponible à la figure 6.3.

Figure 6.3 – ajout d'une politique



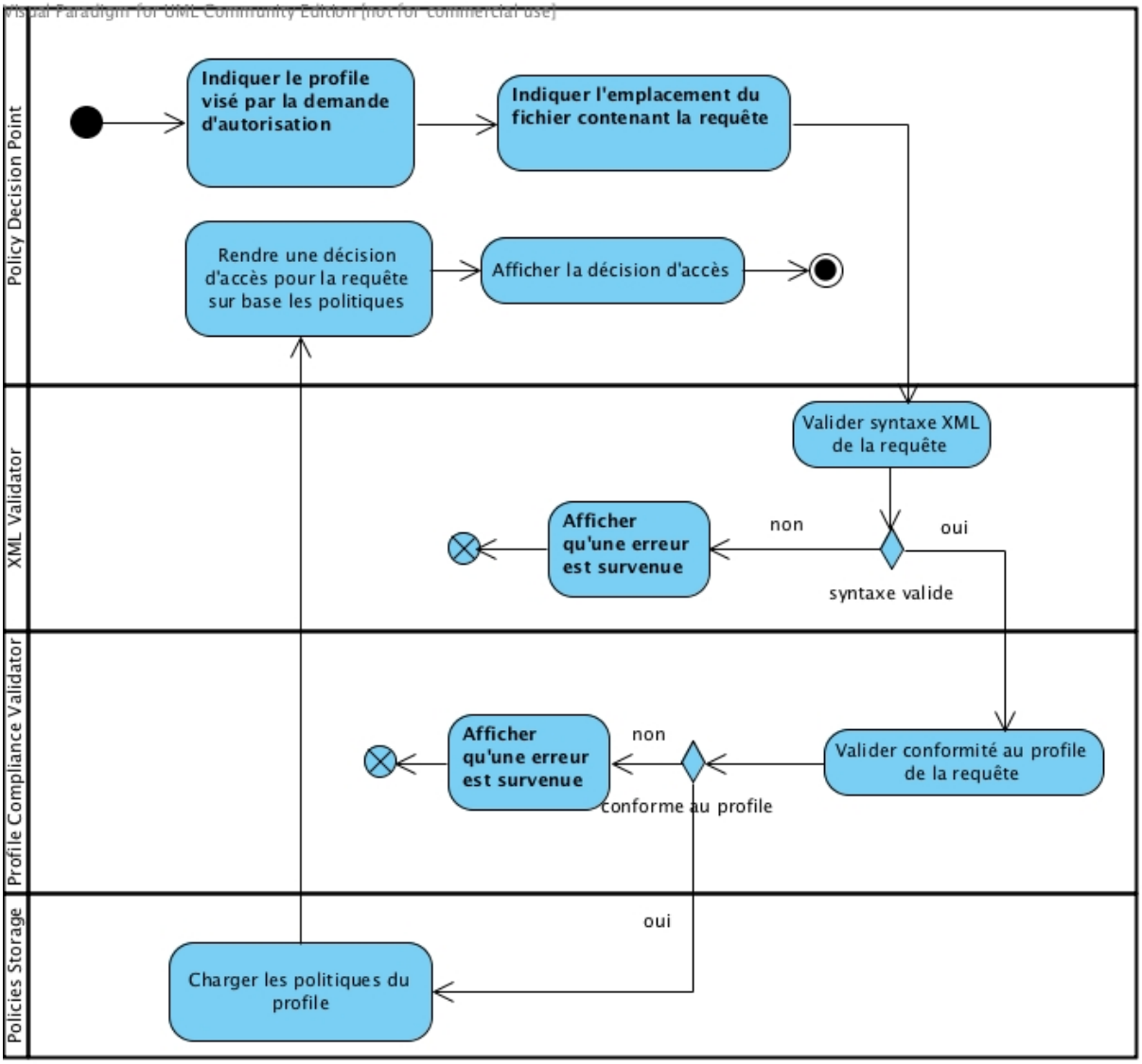
6.4 Évaluation de règles

Enfin, l'évaluation en elle-même des politiques sera précédée d'une traduction de celles-ci en XACML, si elles sont décrites dans un langage autre que XACML.

6.4.1 Requête pour un profil XACML

Le diagramme d'activités correspondant à une requête pour un profil XACML est disponible à la figure 6.4.

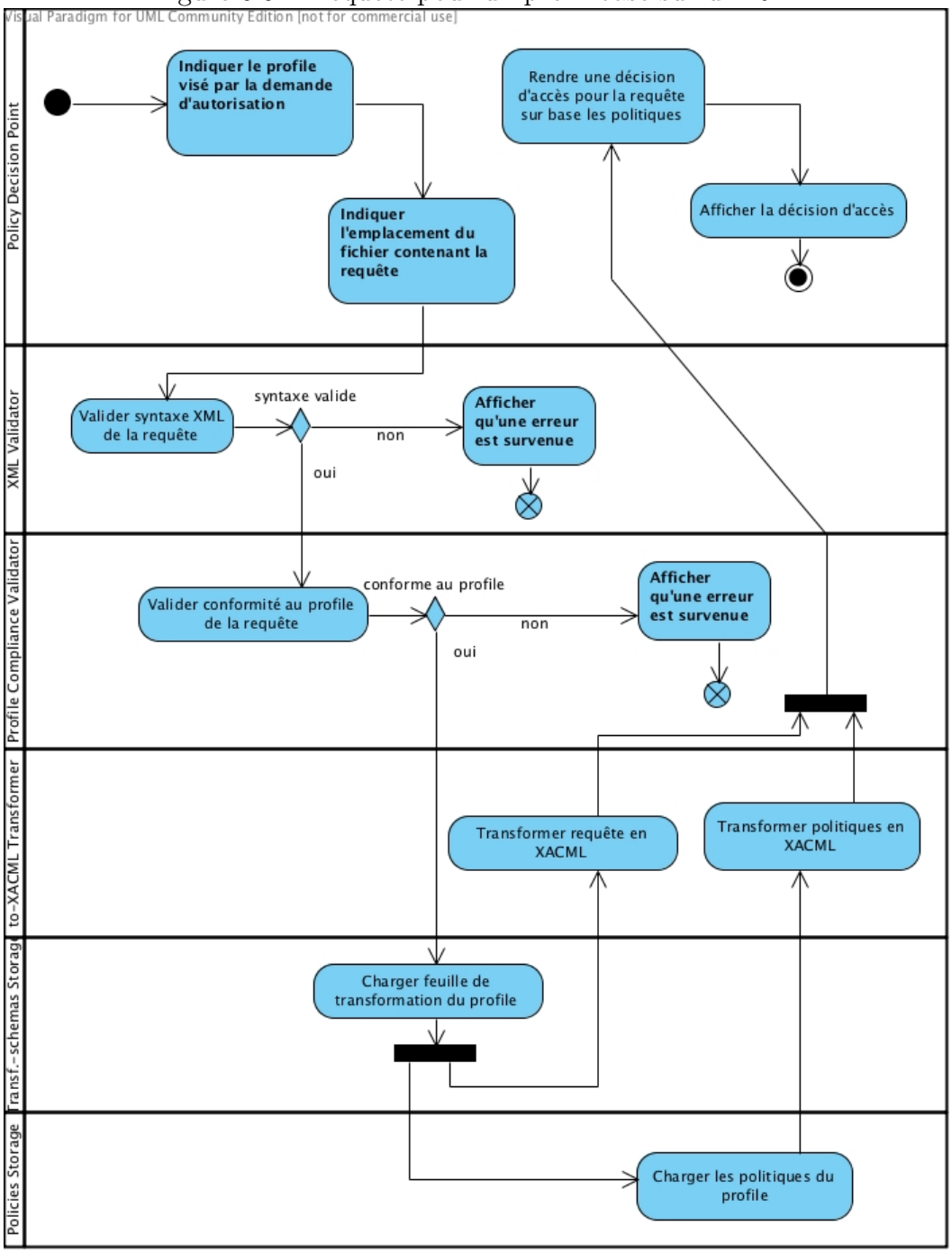
Figure 6.4 – requête pour un profil XACML



6.4.2 Requête pour un profil basé sur un REL

Le diagramme d'activités correspondant à une requête pour un profil basé sur un REL est disponible à la figure 6.5.

Figure 6.5 – requête pour un profil basé sur un REL



CHAPITRE 7

CAS D'UTILISATION

Dans ce troisième et dernier chapitre de la deuxième partie de ce travail, nous présentons trois cas concrets d'utilisation du prototype de moteur d'évaluation de règles, montrant les possiblités offertes par celui-ci en terme de généricité et d'ajout de types de règles évaluables.

Il est cependant important de préciser que ces cas restent des cas basiques, qui n'utilisent pas pleinement les ressources d'expressivité offertes par le langage XACML.

De plus, à l'aide des langages d'expression de droits (ou autres types de langages d'expression de politiques), des profils beaucoup plus complexes et fournis en nombre d'entités utilisées et de contraintes additionnelles définies peuvent être décrits et intégrés au prototype, en suivant les exigences et principes d'utilisation de celui-ci donnés aux chapitre 5 et 6.

7.1 Modèle R-BAC en XACML

Le modèle R-BAC, qui a été présenté à la section 2.4 de ce mémoire et dont le profil pour XACML a été décrit à la section 3.1.1.2, constitue un des modèles de contrôle d'accès majeurs et son profil pour XACML est un des profils de base défini pour ce langage.

7.1.1 Constitution du schéma du profil

Nous reprenons dans le tableau 7.I les principales correspondances entre les entités définies par le modèle de contrôle d'accès et leurs représentations en XACML telles qu'elles ont été déterminées dans la définition du profil [26] par le comité de spécification OASIS.

Tableau 7.I – représentation XACML des éléments du modèle R-BAC

Élément	**Représentation XACML**
rôle	Subject Attribute
définition d'un rôle	Role <PolicySet> avec `PolicySetId="RPS:role1:role"`
définition des permissions associées à un rôle	Permission <PolicySet> avec `PolicySetId="PPS:role1:role"`
attribution d'un rôle à des utilisateurs	Role Assignement <Policy> avec `PolicyId="Role:Assignment:Policy"`

Suivant les principes pour la définition de schémas compatibles avec le prototype définis au chapitre 6, nous considérons tout d'abord l'ensemble des éléments des schémas du langage XACML comme éléments potentiels du schéma final du profil.

Pour les étapes 2, 3 et 4 de la définition d'un schéma compatible, le profil R-BAC pour XACML utilisant les entités de base du langage sans ajouter de restrictions complémentaires, nous conservons comme partie intégrante du profil tous les éléments des schémas du langage XACML, avec leurs cardinalités et définitions d'origine.

Comme les éléments repris dans le tableau 7.I permettent de le voir, aucun élément supplémentaire n'est requis pour la définition des entités du profil et ne doit donc être ajouté au schéma du langage. De ce fait, les **schémas d'origine du langage XACML**, sont, sans aucune modification, les schémas du profil R-BAC. Les fichiers XML contenant ces schémas sont disponibles à l'annexe A.

Enfin, ce profil R-BAC étant exprimé en utilisant le langage natif du moteur d'évaluation de règles, **aucun schéma de transformation vers XACML n'est requis** ici.

Une fois le schéma du profil décrit, celui-ci peut être ajouté, par importation, aux profils reconnus par le prototype et contenus dans la base de connaissances.

7.1.2 Ajout de politiques d'exemple au prototype

Le profil R-BAC pour XACML a la particularité, provenant de la nature même du modèle de contrôle d'accès défini sur des rôles (faisant le lien entre les permissions offertes et les utilisateurs du système), de recourir à plusieurs entités *Policy* exprimant différents concepts pour décrire complétement une politique. Ces différentes utilisations de *Policy* sont celles reprises dans le tableau 7.I.

Pour créer une politique d'exemple, nous définissons donc tout d'abord deux rôles *professor* et *student* (exprimés chacun par une **Role <Policy>** et d'identifiants respectifs "RPS :professor :role" et "RPS :student :role").

Les permissions *create* et *join* sur une ressource *webcampus-group* sont décrites chacune par une **Permission <Policy>** respectivement associée au rôle *professor* et *student*.

Enfin, pour l'assignation d'utilisateurs aux différents rôles afin de pouvoir actionner des sessions d'utilisation, nous complétons cette description de politique par la définition d'une **Role Assignement <Policy>**. Le rôle *student* est assigné à Alice et Bob, pour une utilisation uniquement entre 8 heures et 19 heures, le rôle *professor* est assigné à Carole, sans condition.

L'ensemble des éléments faisant partie de la définition de la politique d'exemple sont repris à l'annexe B. Ceux-ci, contenus dans des fichiers XML, peuvent être importés dans le prototype au sein des politiques de la base de connaissances.

7.1.3 Évaluation de requêtes sur les politiques d'exemple

Nous décrivons ici une requête d'accès XACML par utilisateur, afin de tester que son rôle défini lui est bien attribué. Basiquement, nous avons donc trois requêtes,

une pour Bob, une pour Alice et une pour Carole. Elles sont applicables sur la **Role Assignement <Policy>**, qui définit l'attribution des rôles.

Les requêtes, sous forme de fichiers XML, sont également disponibles à l'annexe B. Elles peuvent être testées sur le prototype et seront applicables si les politiques définies à la section précédente sont bien contenues dans la base de connaissances.

7.2 Licences Creative Commons en ODRL 2.0

Le profil Creative Commons pour ODRL 2.0, dont l'essai de spécification est disponible en annexe I de ce mémoire, constitue un exemple simple d'utilisation du langage ODRL pour la description de licences sur des biens divers.

7.2.1 Constitution du schéma du profil

Suivant de nouveau les principes pour la définition de schémas compatibles avec le prototype définis au chapitre 6, nous démarrons la constitution du schéma du profil en considérant l'ensemble des éléments du modèle ODRL 2.0 [16].

Comme pour le profil R-BAC pour XACML, tous les éléments du modèle du langage de base sont conservés. En effet, le modèle ODRL 2.0 fournit une structure simplifiée mais difficilement réductible avec un nombre d'entités maîtrisé.

Cependant, dans l'exemple présent d'utilisation du prototype, nous restreignons quelque peu certaines entités afin de limiter les contours du profil. Ainsi, les types `conflictENUM` et `undefinedENUM` ainsi que les attributs de l'entité *policy* `conflict`, `undefined`, `inheritAllowed`, `inheritFrom` et `inheritRelation` sont retirés du schéma du profil. De plus, seules les types de *policy* `offer`, `agreement` et `request` sont conservés.

Enfin, un sous-ensemble des *actions* et *constraints* du vocabulaire associé au langage est considéré pour transformation en XACML dans la section suivante.

De leur côté, les cardinalités des entités restent les mêmes et tous les éléments du vocabulaire du langage peuvent être utilisés. L'unique ajout de terme prévu au dictionnaire du profil est une recommandation d'utiliser l'URL "`http://creativecommons.org/ns#HighIncomeNationUse`" pour représenter la prohibition *HighIncomeNationUse* dans le cas où la licence utilisée exclurait de jouir des droits sur la ressource visée dans des pays définis comme étant "*à hauts revenus*" par la Banque Mondiale [1].

Le schéma du profil tel qu'il peut être utilisé pour ajout au prototype est disponible à l'annexe C.

7.2.2 Constitution du schéma de transformation vers XACML

Pour ce second cas d'exemple d'utilisation du prototype, le langage supportant le profil n'étant pas XACML mais ODRL, il est nécessaire de définir un schéma de transformation qui propose des correspondances en éléments XACML pour l'ensemble des entités du schéma du profil.

Comme cela été indiqué au chapitre 6, aucune méthode n'est fournie. La définition du schéma de transformation au format XSL doit naturellement être réalisée en accord avec les spécificités du profil et du langage visé.

Dans le cas du profil Creative Commons pour ODRL, il faut particulièrement prendre en compte chaque valeur possible pour les noms d'*actions* (`commercialize`, `copyReproduce`, `distribute`, `deriveModify`, `share` et `highIncomeNationUse`) propres au domaine et qui sont utilisées, selon les cas, au sein d'entités *permission*, *prohibition* ou *duty*. Après transformation, ces éléments représentent les *Actions* au sein de *Rules* XACML.

Le schéma de transformation associé au profil et reprenant l'ensemble des éléments mentionnés ici est disponible à l'annexe D de ce mémoire.

[1] `http://data.worldbank.org/about/country-classifications`

7.2.3 Ajout de politiques d'exemple au prototype

Pour la description d'une politique d'exemple à ajouter au prototype, nous reprenons un cas fictif de chanteuse, Anita, souhaitant distribuer un morceau de musique de sa composition. Cet exemple avait à l'origine été décrit sur le site du projet Creative Commons, puis a été repris comme exemple lors de la définition du profil Creative Commons pour ODRL 1.1. Enfin, il a également été utilisé lors de l'essai de spécification de profil Creative Commons pour ODRL 2.0 disponible à l'annexe I de ce mémoire.

Deux politiques sont définies. La première décrit une licence *Attribution-NonCommercial-ShareAlike* sur le morceau de musique d'Anita, identifié par l'URN "`urn:anita-music:songs:volcanoLove.mp3`" alors que la seconde licence qu'Anita décrit en ODRL concerne un document reprenant les paroles de son morceau de musique. Pour celui-ci, elle définit une licence *Attribution-NonCommercial-NoDerivs* et le document est identifié par l'URN "`urn:anita-music:lyrics:volcanoLove.pdf`".

Pour chaque licence, un utilisateur a accepté les termes de la licence et est donc libre d'utiliser le bien dont les droits sont décrits. Ainsi Bob a accepté la première licence décrivant les droits sur le morceau de musique tandis qu'Alice a accepté les termes de la seconde licence pour les paroles du morceau de musique et peut donc y accéder sans contrainte. Ces utilisateurs sont mentionnés au sein de chaque licence en tant que *party* avec une fonction d'*assignee*. Ces deux licences sont de fait des *agreements*.

Les fichiers XML reprenant l'ensemble de ces politiques sont repris à l'annexe E de ce mémoire.

7.2.4 Évaluation de requêtes sur les politiques d'exemple

Afin de réaliser des requêtes sur ces licences, le type de *policy* ODRL `request` est utilisé. Ainsi, une requête pour chaque licence est décrite, permettant de tester si un sujet possède l'autorisation de réaliser une action sur la ressource voulue.

La première requête porte sur le morceau de musique pour lequel Anita a décidé de définir une licence Creative Commons. Cette requête va vérifier que Bob, qui est un utilisateur pour qui des permissions ont été définies, est bien autorisé à reproduire (action `copyReproduce`) le travail d'Anita. La seconde requête permet quand à elle de vérifier qu'Alice possède bien la permission d'imprimer (action `print`) le document contenant les paroles de la chanson d'Anita.

Ces deux requêtes sont disponibles à l'annexe E. Elles peuvent être testées sur le prototype et seront applicables si les politiques définies à la section précédente sont bien contenues dans la base de connaissances.

7.3 Mobile Profile en MPEG-21 REL

7.3.1 Constitution du schéma du profil

Reprenant une dernière fois la procédure décrite au chapitre 6 pour la constitution de schémas compatibles avec le prototype, nous considérons l'ensemble des éléments du langage MPEG-21 REL. Celui-ci étant défini notamment en terme d'extensions [38], nous considérons ici les trois entités principales, à savoir le *Core Language*, la *Standard Extension* et la *Multimedia Extension*.

Le profil que nous prenons ici comme exemple porte sur un domaine relativement restreint [35] et certains éléments des trois schémas de base du langage peuvent être ignorés ou limités. Les entités *principal* et *resource* sont notamment réduites à une seule de leur valeur possible, spécifique à une utilisation sur des appareils mobiles. De cette manière, ce sont respectivement et uniquement des *keyHolder* et

des *diReference* qui décriront les (clés uniques des) utilisateurs et les ressources au sein des licences.

Les *conditions* pouvant être utilisées sont également limitées pour la définition de ce profil. Seules trois conditions, permettant de contrôler l'usage fait des contenus sur les appareils, sont conservées. La condition *validityInterval* permet de définir des limites temporelles au-delà desquelles l'usage du contenu est exclu. *validityIntervalFloating* et *exerciceLimit* définissent respectivement une durée d'utilisation du contenu et un nombre de fois que celui-ci peut être utilisé. Enfin, l'agrégateur *allConditions* est également conservé afin de pouvoir combiner plusieurs de ces conditions au sein d'une même définition de droit.

Enfin, trois valeurs de *right* sont considérées, définissant les actions potentiellement attribuables sur les contenus : *play*, *print* et *execute*.

Dans un souci de simplicité, l'ensemble des entités et valeurs faisant partie du profil sont regroupées au sein d'un même schéma XSD unifié. Il existe donc un schéma unique du profil. Celui-ci peut être trouvé à l'annexe F de ce mémoire.

7.3.2 Constitution du schéma de transformation vers XACML

Dans un second temps et une fois le schéma du profil décrit, nous constituons le schéma de transformation vers XACML.

Plusieurs difficultés se posent lors de cette étape. Tout d'abord, le langage MPEG-21 REL ne définissant pas d'entité pour exprimer des requêtes, il est nécessaire, tout en respectant la sémantique du langage, de trouver un mécanisme permettant d'interroger des licences (c'est-à-dire vérifier qu'un utilisateur possède la permission d'effectuer une opération donnée sur une ressource particulière). Une première solution est de ne pas intégrer de mécanisme de requêtes directement au schéma du profil mais simplement d'écrire celles-ci en XACML. En effet, l'évaluation des politiques est réalisée sur les versions transformées en XACML des licences et les

requêtes sont également préalablement transformées en XACML. Cependant, cette solution paraît inadaptée, puisqu'elle demande à des utilisateurs du prototype définissant des licences grâce au langage MPEG-21 REL d'avoir recours à un autre langage d'expression de politiques.

Une seconde solution est de définir un mécanisme de requêtes directement au sein du langage MPEG-21 REL. Cela demande de respecter la sémantique du langage et donc de limiter l'ajout de nouvelles entités ou valeurs à des termes allant dans le sens orginal du langage. Cette solution est utilisée ici de la manière suivante : une requête est exprimée sous forme d'une licence possédant un attribut `licenceId` avec une valeur débutant par "request". De plus, la définition de la licence ne contient que les éléments nécessaires à l'évaluation, à savoir, au sein d'une entité *grant*, un *principal*, un *right* et une *resource*.

Une seconde difficulté réside dans le choix des mécanismes XACML à utiliser pour représenter les *conditions*. En effet, les possibilité offertes par XACML sont très variées et permettent des représentations très diverses. Pour l'expression de la condition, *validityInterval* suit pour sa part la définition d'une condition souvent exploitée, décrivant la jonction de deux contraintes (plus grand ou égal et plus petit ou égal) prenant comme valeur des dates fixes et à comparer avec la date actuelle.

Les conditions *validityIntervalFloating* et *exerciceLimit* sont, elles, définies pour comparer deux valeurs et renvoyer un résultat positif si la première valeur est supérieure à la seconde. Les deux valeurs à comparer sont la durée pendant laquelle l'action considérée a déjà été réalisée sur le contenu et zéro. Le premier élément utilisé par cette condition doit être mis à jour dans l'expression de la licence lorsque sa valeur change. Pour la condition *exerciceLimit*, le même mécanisme de comparaison est appliqué, la valeur à comparer à zéro étant simplement le nombre de fois que l'action a déjà été réalisée sur le contenu.

Le schéma de transformation complet, reprenant l'ensemble des éléments mention-

nés, est disponible à l'annexe G de ce mémoire.

7.3.3 Ajout de politiques d'exemple au prototype

Les politiques d'exemple utilisées dans cette section sont reprises du document de ContentGuard définissant le profil [35]. Seules quelques modifications mineures ont été apportées (ajout d'une condition et unification des valeurs d'appareils utilisateur et de distributeur utilisés).

Les deux licences exprimées reprennent des cas classiques d'expression de droits sur des contenus numériques à destination des appareils mobiles. Les appareils sont identifiés par leur clé unique d'identification [15]. Les distributeurs sont également identifiés par une clé signée. La première licence décrite permet à un appareil "`P800`" d'éxecuter une fois une application "`urn:mobile:newsReader.app`". La seconde licence, attribuée à un appareil "`PDA-Model-300`", donne la permission à l'utilisateur de jouer un contenu "`travel:guides:ny:centralpark`" et d'en imprimer des informations contenues une unique fois.

La fichiers reprenant les descriptions de ces deux licences sont disponibles à l'annexe H de ce mémoire.

7.3.4 Évaluation de requêtes sur les politiques d'exemple

Les requêtes qui sont définies pour la vérification de ces licences visent particulièrement à analyser, pour l'appareil "`P800`" sur le contenu "`urn:mobile:newsReader.app`", la permission d'exécuter celui-ci et pour l'appareil "`PDA-Model-300`" sur le contenu "`travel:guides:ny:centralpark`" la permission de jouer celui-ci.

Ces deux requêtes, décrites en respectant la méthode choisie expliquée à la section 7.3.2, sont disponibles à la fin de l'annexe H de ce mémoire.

CHAPITRE 8

CONCLUSION

8.1 Réponses aux questions de recherche

Trois questions de recherche ont été introduites au début de ce travail. Elles ont pour vocation de structurer mais également de conclure ce mémoire en apportant des réponses concrètes. Ces réponses, dont le détail est donné tout au long de ce mémoire, sont résumées ici.

Quelle est la meilleure manière de définir des règles de sécurité et d'exprimer des contraintes ? (Q1)
L'analyse des modèles de contrôle d'accès réalisée au chapitre 2 fait ressortir l'importance de considérer les particularités du domaine d'application visé et de choisir précisément les critères de sélection du modèle le plus adéquat. La diversité des solutions proposées permet aujourd'hui de faire face à un nombre conséquent de situation potentielles.

Plus encore, certains modèles, très à même de rencontrer les contraintes des organisations complexes et étant basés sur des notions génériques, font l'objet de multiples tentatives d'extension. Ainsi, le modèle R-BAC est à la base de nombre de déclinaisons mises en place pour des domaines spécifiques. Le modèle ABAC permet, par sa structure centrée sur la définition d'attributs, de mettre en oeuvre d'autres modèles de contrôle d'accès. Le standard XACML repose de plus sur son modèle simple.

Pour le second aspect inhérent à la définition de règles de contrôle d'accès, à savoir l'expression de celles-ci, la palette des solutions existantes est également très vaste. Cependant, comme nous l'avons vu au chapitre 3 puis au chapitre 4, des langages majeurs ressortent. Le premier d'entre eux, XACML, est celui qui offre la plus

grande expressibilité. Il est d'ailleurs un standard reconnu et le fait que soit défini dans sa spécification un protocole formel de requêtes et réponses d'autorisation ajoute à son intérêt. Dans le domaine plus spécifique des langages d'expression de droits, ODRL, et notamment dans sa seconde version, fait de plus en plus office de standard ouvert pour l'expression de droits numériques. L'alternative MPEG-21 REL utilise des entités qui rencontrent moins facilement les entités du monde réel mais reste une norme reconnue et dont des sous-ensembles (tel OMA REL) servent de base à la gestion des droits pour des biens très divers.

Quelles technologies sont aptes à supporter de manière efficace l'évaluation automatique de politiques ? (Q2)

Chaque langage possédant ses caractérisitques d'expression propres et chaque domaine d'application possédant des contraintes différentes, de multiples mécanismes d'évaluation automatique de politiques se côtoient. Les solutions offertes par XACML et par EPAL analysées au chapitre 3, autant dans les normes communes utilisées [12] que dans la diversité des implémentations existantes pour XACML, montrent que la tendance est à la définition de standards décrivant des méthodes pouvant être mises en place dans les situations les plus diverses.

Les tentatives d'interopérabilité entre langages d'expression de droits numériques décrites à la fin du chapitre 4 vont, de ce point de vue, dans le même sens. Elles permettent de se rendre compte de la difficulté d'unifier des expressions de droits aux caractéristiques (langage d'expression utilisé, domaine d'application visé) parfois très différentes.

La conception du prototype de moteur d'évaluation de règles dont les choix sont décrits au chapitre 5, mise sur le langage XACML et son expressivité pour fournir une solution d'évaluation efficace grâce à un module unique supportant la transformation de profils de langages en des politiques XACML. Afin de préserver une simplicité d'appréhension et d'utilisation, une méthode générale pour la description de profils compatibles avec le prototype est décrite. Les exemples donnés au

chapitre 7 pour l'intégration au prototype de nouveaux types de règles suivent les étapes de cette méthode pour l'ajout de nouveaux profils.

Enfin, le recours quasi-exclusif à des technologies XML pour la description des schémas des profils, l'expression des politiques et la définition des transformations vers XACML laisse la porte ouverte à l'ajout potentiel d'un grand nombre de types de règles, ces technologies étant très majoritairement utilisées pour l'expression de politiques de sécurité (et particulièrement l'expression de droits numériques).

Comment adopter une approche générique dans la construction d'une architecture d'évaluation automatique de politiques ? (Q3)
La difficulté de répondre à une contrainte de généricité réside autant dans la désignation des mécanismes capables de combiner plusieurs langages en vue d'une évaluation unique que dans la mise en place efficace du mécanisme finalement sélectionné. Les choix qui ont été faits dans la conception apportent des solutions à plusieurs niveaux, tout en veillant à ne pas fournir un résultat final dont la complexité empêcherait des ouvertures futures.

Le choix de XACML comme langage natif d'évaluation est sans doute le plus évident et celui qui s'impose le plus facilement lorsque l'on souhaite mettre en place une architecture générique d'évaluation de politiques. D'autres technologies telles que celles imaginées par *JBoss* au sein de la suite Drools [21] peuvent aussi répondre à ces attentes, cependant au prix du recours à un paradigme déclaratif pour exprimer les contraintes inhérentes aux langages considérés.

La définition d'une approche particulière pour l'ajout du support de nouveaux profils d'utilisation au prototype (en une seule phase, considérant uniquement des profils de langages) contribue à favoriser la généricité des types de règles acceptés et l'extensibilité future à de nouveaux types. Cette approche requiert en effet uniquement la définition des transformations vers XACML des entités des langages faisant partie des schémas potentiellement restreints des profils. Le fonctionnement

précis du prototype est décrit au chapitre 6.

8.2 Travaux futurs

Ce travail a été l'occasion d'une analyse du domaine des langages d'expression des politiques de sécurité et notamment des moyens de mettre en oeuvre une évaluation automatique efficace de ces politiques.

Cependant, certaines contraintes, temporelles ou inhérentes à la définition d'un cadre restreint pour le travail, laissent la porte ouverte à des améliorations ou travaux futurs.

Au niveau de l'évaluabilité des politiques XACML provenant de transformations, une analyse plus poussée pourrait permettre de réaliser cette tâche de manière automatisée et systématique. De plus, le prototype fourni permet potentiellement l'ajout du support de nombreux langages. Les tentatives d'intégration ont été limitées à des profils définis sur les trois langages XACML, ODRL et MPEG-21 REL. Ceux-ci sont des langages majeurs mais d'autres langages, peut-être plus restrictifs ou méconnus, pourraient également faire l'objet d'intégration au prototype.

Enfin, l'implémentation du prototype limite le cadre des langages potentiellement candidats à l'intégration aux langages pouvant être décrits à l'aide d'une syntaxe XML. Une approche visant à transformer des descriptions exprimées dans une syntaxe différente en entités du langage XACML pourrait être envisagée, profitant une nouvelle fois de l'expressivité très forte de ce standard.

Au-delà d'une approche syntaxe, d'autres modèles de représentation de politiques, comme les langages de contrôle d'accès sémantique [14] [24], pourraient être analysés afin d'évaluer si leur intégration dans un système unifié d'évaluation de politiques serait envisageable.

BIBLIOGRAPHIE

[1] Diala Abi Haidar, Nora Cuppens-Boulahia, Frederic Cuppens, and Herve Debar. An extended rbac profile of xacml. In *Proceedings of the 3rd ACM workshop on Secure web services*, SWS '06, pages 13–22, New York, NY, USA, 2006. ACM.

[2] Open Mobile Alliance. Oma digital rights management, 2008.

[3] Christopher Alm and Roland Illig. Translating high-level authorization constraints to xacml. In *Proceedings of the 2010 6th World Congress on Services*, SERVICES '10, pages 629–636, Washington, DC, USA, 2010. IEEE Computer Society.

[4] Anne Anderson. A comparison of two privacy policy languages : Epal and xacml. Technical report, Mountain View, CA, USA, 2005.

[5] Elliott Bell and Leonard LaPadula. Secure computer systems : Mathematical foundations", 1973.

[6] J. K. Biba. Integrity considerations for secure computer systems, 1977.

[7] Fei Chen and Alex X. Liu. Xengine : A fast and scalable xacml policy evaluation engine. 2008.

[8] International Press Telecommunications Council. Specification of a profile and vocabulary for the communication of usage rights in odrl 2.0. 2012.

[9] Yuri Demchenko, Mihai Cristea, and Cees de Laat. Xacml policy profile for multidomain network resource provisioning and supporting authorisation infrastructure. In *Proceedings of the 2009 IEEE International Symposium on Policies for Distributed Systems and Networks*, POLICY '09, pages 98–101, Washington, DC, USA, 2009. IEEE Computer Society.

[10] David Ferraiolo and Vijay Atluri. A meta model for access control : why is it needed and is it even possible to achieve ? In *Proceedings of the 13th ACM symposium on Access control models and technologies*, SACMAT '08, pages 153–154, New York, NY, USA, 2008. ACM.

[11] David Ferraiolo and Rick Kuhn. Role-based access controls. *15th National Computer Security Conference*, 1992.

[12] International Organization for Standardization. Iso/iec 10181-3 :1966 information technology – open systems interconnection – security frameworks for open systems : Access control framework. Technical report.

[13] John Robert Gardner and James Robert Gardner. *XSLT and XPATH : A Guide to XML Transformations*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[14] Tonti Gianluca, Bradshaw Jeffrey, Jeffers Renia, Montanari Rebecca, Suri Niranjan, , and Uszok Andrzej. Semantic web languages for policy representation and reasoning : A comparison of kaos, rei, and ponder. In *ISWC 2003 : international semantic web conference No2*, 2003.

[15] Jun Gong and Peter Tarasewich. Alphabetically constrained keypad designs for text entry on mobile devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, pages 211–220, New York, NY, USA, 2005. ACM.

[16] W3C ODRL Community Group. Odrl version 2.0 core model. 2012.

[17] Matthew Harren, Mukund Raghavachari, Oded Shmueli, Michael G. Burke, Rajesh Bordawekar, Igor Pechtchanski, and Vivek Sarkar. Xj : facilitating xml processing in java. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 278–287, New York, NY, USA, 2005. ACM.

[18] Renato Iannella. Open digital rights language (odrl). 2002.

[19] Renato Iannella. Odrl creative commons profile, 2005.

[20] JBOSS. Picketbox xacml, 2011.

[21] JBOSS. Drools expert, 2012.

[22] Polo Josep, Prados Jose, and Delgado Jaime. Interoperability between odrl and mpeg-21 rel. In *Proceedings of the First International ODRL Workshop (Eds. Renato Iannella and Susanne Guth)*, 2004.

[23] Hogab Kang, Keunyoung Lee, Taehyun Kim, Xin Wang, and Jaime Delgado. Interoperability between the mpeg-21 rel dac profile and other rights information standards. In *Proceedings of the Third International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, AXMEDIS '07, pages 155–162, Washington, DC, USA, 2007. IEEE Computer Society.

[24] Yague M.I. and Troya J.M. A semantic approach for access control in web services. In *EuroWeb 2002 Conference*, 2002.

[25] Qun Ni, Alberto Trombetta, Elisa Bertino, and Jorge Lobo. Privacy-aware role based access control. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 41–50, New York, NY, USA, 2007. ACM.

[26] OASIS. Xacml profile for role based access control (rbac) version 1.0, February 2004.

[27] OECD. Oecd guidelines on the protection of privacy and transborder flows of personal data, 1980.

[28] Eva Rodriguez and Jaime Delgado. Towards the interoperability between mpeg-21 rel and creative commons licenses. In *Proceedings of the Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, AXMEDIS '06, pages 45–52, Washington, DC, USA, 2006. IEEE Computer Society.

[29] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, Boston, MA, 2. edition, 2005.

[30] Inc. Sun Microsystems. Sun's xacml, 2004.

[31] W3C. Xpath specification, 1999.

[32] W3C. Dtd specification, 2004.

[33] W3C. Xsd specification, 2004.

[34] W3C. Xsl specification, 2008.

[35] Xin Wang. Mpeg-21 rights expression language : Enabling interoperable digital rights management. *IEEE MultiMedia*, 11(4) :84–87, October 2004.

[36] Xin Wang, Guillermo Lao, Thomas DeMartini, Hari Reddy, Mai Nguyen, and Edgar Valenzuela. Xrml - extensible rights markup language. In *Proceedings of the 2002 ACM workshop on XML security*, XMLSEC '02, pages 71–79, New York, NY, USA, 2002. ACM.

[37] Maronas Xavier, Rodriguez Eva, and Delgadol Jaime. An architecture for the interoperability between rel based on xacml. 2007.

[38] Wang Xin, DeMartini Thomas, Wragg Barney, Paramasivam M., and Barlas Chris. The mpeg-21 rights expression language and rights data dictionary. 2003.

[39] Min Xu and Duminda Wijesekera. A role-based xacml administration and delegation profile and its enforcement architecture. In *Proceedings of the 2009 ACM workshop on Secure web services*, SWS '09, pages 53–60, New York, NY, USA, 2009. ACM.

[40] Eric Yuan and Jin Tong. Attributed based access control (abac) for web services. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '05, pages 561–569, Washington, DC, USA, 2005. IEEE Computer Society.

[41] Yi Zhang, Xiaoguang Wei, and Shu Zhao. Research on odrl connecting with application scenarios. In *Proceedings of the 2008 International Seminar on Future Information Technology and Management Engineering*, FITME '08, pages 261–264, Washington, DC, USA, 2008. IEEE Computer Society.

ANNEXE A

SCHÉMA XML DU PROFILE R-BAC POUR XACML

Listing A.1 – xacml-policy-schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:oasis:names:tc:xacml:1.0:policy"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xacml="
    urn:oasis:names:tc:xacml:1.0:policy" elementFormDefault="qualified
    " attributeFormDefault="unqualified">
  <!-- -->
  <xs:element name="PolicySet" type="xacml:PolicySetType"/>
  <xs:complexType name="PolicySetType">
    <xs:sequence>
      <xs:element ref="xacml:Description" minOccurs="0"/>
      <xs:element ref="xacml:PolicySetDefaults" minOccurs="0"/>
      <xs:element ref="xacml:Target"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="xacml:PolicySet"/>
        <xs:element ref="xacml:Policy"/>
        <xs:element ref="xacml:PolicySetIdReference"/>
        <xs:element ref="xacml:PolicyIdReference"/>
      </xs:choice>
      <xs:element ref="xacml:Obligations" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
    <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="
        required"/>
  </xs:complexType>
  <!-- -->
  <xs:element name="PolicySetIdReference" type="xs:anyURI"/>
  <xs:element name="PolicyIdReference" type="xs:anyURI"/>
  <!-- -->
  <xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
  <xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
```

```
<xs:complexType name="DefaultsType">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="xacml:XPathVersion"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="XPathVersion" type="xs:anyURI"/>
<!-- -->
<xs:element name="Policy" type="xacml:PolicyType"/>
<xs:complexType name="PolicyType">
  <xs:sequence>
    <xs:element ref="xacml:Description" minOccurs="0"/>
    <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
    <xs:element ref="xacml:Target"/>
    <xs:element ref="xacml:Rule" minOccurs="0" maxOccurs="unbounded"
        />
    <xs:element ref="xacml:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
  <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="
      required"/>
</xs:complexType>
<!-- -->
<xs:element name="Description" type="xs:string"/>
<!-- -->
<xs:element name="Rule" type="xacml:RuleType"/>
<xs:complexType name="RuleType">
  <xs:sequence>
    <xs:element ref="xacml:Description" minOccurs="0"/>
    <xs:element ref="xacml:Target" minOccurs="0"/>
    <xs:element ref="xacml:Condition" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="RuleId" type="xs:anyURI" use="required"/>
```

```
    <xs:attribute name="Effect" type="xacml:EffectType" use="required"
        />
61  </xs:complexType>
    <!-- -->
    <xs:simpleType name="EffectType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="Permit"/>
66      <xs:enumeration value="Deny"/>
      </xs:restriction>
    </xs:simpleType>
    <!-- -->
    <xs:element name="Target" type="xacml:TargetType"/>
71  <xs:complexType name="TargetType">
      <xs:sequence>
        <xs:element ref="xacml:Subjects"/>
        <xs:element ref="xacml:Resources"/>
        <xs:element ref="xacml:Actions"/>
76    </xs:sequence>
    </xs:complexType>
    <!-- -->
    <xs:element name="Subjects" type="xacml:SubjectsType"/>
    <xs:complexType name="SubjectsType">
81    <xs:choice>
        <xs:element ref="xacml:Subject" maxOccurs="unbounded"/>
        <xs:element ref="xacml:AnySubject"/>
      </xs:choice>
    </xs:complexType>
86  <!-- -->
    <xs:element name="Subject" type="xacml:SubjectType"/>
    <xs:complexType name="SubjectType">
      <xs:sequence>
        <xs:element ref="xacml:SubjectMatch" maxOccurs="unbounded"/>
91    </xs:sequence>
    </xs:complexType>
    <!-- -->
    <xs:element name="AnySubject"/>
```

```
<!-- -->
<xs:element name="Resources" type="xacml:ResourcesType"/>
<xs:complexType name="ResourcesType">
  <xs:choice>
    <xs:element ref="xacml:Resource" maxOccurs="unbounded"/>
    <xs:element ref="xacml:AnyResource"/>
  </xs:choice>
</xs:complexType>
<!-- -->
<xs:element name="AnyResource"/>
<!-- -->
<xs:element name="Resource" type="xacml:ResourceType"/>
<xs:complexType name="ResourceType">
  <xs:sequence>
    <xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Actions" type="xacml:ActionsType"/>
<xs:complexType name="ActionsType">
  <xs:choice>
    <xs:element ref="xacml:Action" maxOccurs="unbounded"/>
    <xs:element ref="xacml:AnyAction"/>
  </xs:choice>
</xs:complexType>
<!-- -->
<xs:element name="AnyAction"/>
<!-- -->
<xs:element name="Action" type="xacml:ActionType"/>
<xs:complexType name="ActionType">
  <xs:sequence>
    <xs:element ref="xacml:ActionMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="SubjectMatch" type="xacml:SubjectMatchType"/>
```

```
<xs:complexType name="SubjectMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:SubjectAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="ResourceMatch" type="xacml:ResourceMatchType"/>
<xs:complexType name="ResourceMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:ResourceAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="ActionMatch" type="xacml:ActionMatchType"/>
<xs:complexType name="ActionMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:ActionAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
```

```
<xs:element name="AttributeSelector" type="
    xacml:AttributeSelectorType"/>
<xs:complexType name="AttributeSelectorType">
  <xs:attribute name="RequestContextPath" type="xs:string" use="
      required"/>
  <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
  <xs:attribute name="MustBePresent" type="xs:boolean" use="optional
      " default="false"/>
</xs:complexType>
<!-- -->
<xs:element name="ResourceAttributeDesignator" type="
    xacml:AttributeDesignatorType"/>
<xs:element name="ActionAttributeDesignator" type="
    xacml:AttributeDesignatorType"/>
<xs:element name="EnvironmentAttributeDesignator" type="
    xacml:AttributeDesignatorType"/>
<!-- -->
<xs:complexType name="AttributeDesignatorType">
  <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
  <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
  <xs:attribute name="Issuer" type="xs:string" use="optional"/>
  <xs:attribute name="MustBePresent" type="xs:boolean" use="optional
      " default="false"/>
</xs:complexType>
<!-- -->
<xs:element name="SubjectAttributeDesignator" type="
    xacml:SubjectAttributeDesignatorType"/>
<xs:complexType name="SubjectAttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:AttributeDesignatorType">
      <xs:attribute name="SubjectCategory" type="xs:anyURI" use="
          optional" default="urn:oasis:names:tc:xacml:1.0:subject-
          category:access-subject"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<!-- -->
<xs:element name="AttributeValue" type="xacml:AttributeValueType"/>
<xs:complexType name="AttributeValueType" mixed="true">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"
        maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
<!-- -->
<xs:element name="Function" type="xacml:FunctionType"/>
<xs:complexType name="FunctionType">
  <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="Apply" type="xacml:ApplyType"/>
<xs:element name="Condition" type="xacml:ApplyType"/>
<!-- -->
<xs:complexType name="ApplyType">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="xacml:Apply"/>
    <xs:element ref="xacml:Function"/>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:element ref="xacml:SubjectAttributeDesignator"/>
    <xs:element ref="xacml:ResourceAttributeDesignator"/>
    <xs:element ref="xacml:ActionAttributeDesignator"/>
    <xs:element ref="xacml:EnvironmentAttributeDesignator"/>
    <xs:element ref="xacml:AttributeSelector"/>
  </xs:choice>
  <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
  <!-- Legal types for the first and subsequent operands are defined
      in the accompanying table -->
</xs:complexType>
<!-- -->
<xs:element name="Obligations" type="xacml:ObligationsType"/>
```

```
226  <xs:complexType name="ObligationsType">
       <xs:sequence>
         <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
       </xs:sequence>
     </xs:complexType>
231  <!-- -->
     <xs:element name="Obligation" type="xacml:ObligationType"/>
     <xs:complexType name="ObligationType">
       <xs:sequence>
         <xs:element ref="xacml:AttributeAssignment" maxOccurs="unbounded
             "/>
236    </xs:sequence>
       <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/
           >
       <xs:attribute name="FulfillOn" type="xacml:EffectType" use="
           required"/>
     </xs:complexType>
     <!-- -->
241  <xs:element name="AttributeAssignment" type="
         xacml:AttributeAssignmentType"/>
     <xs:complexType name="AttributeAssignmentType" mixed="true">
       <xs:complexContent mixed="true">
         <xs:extension base="xacml:AttributeValueType">
           <xs:attribute name="AttributeId" type="xs:anyURI" use="
               required"/>
246      </xs:extension>
       </xs:complexContent>
     </xs:complexType>
     <!-- -->
   </xs:schema>
```

Listing A.2 – xacml-context-schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:oasis:names:tc:xacml:1.0:context"
    xmlns:xacml="urn:oasis:names:tc:xacml:1.0:policy" xmlns:xacml-
    context="urn:oasis:names:tc:xacml:1.0:context" xmlns:xs="http://
```

```
  www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  attributeFormDefault="unqualified">
<xs:import namespace="urn:oasis:names:tc:xacml:1.0:policy"
    schemaLocation="cs-xacml-schema-policy-01.xsd"/>
<!-- -->
<xs:element name="Request" type="xacml-context:RequestType"/>
<xs:complexType name="RequestType">
  <xs:sequence>
    <xs:element ref="xacml-context:Subject" maxOccurs="unbounded"/>
    <xs:element ref="xacml-context:Resource"/>
    <xs:element ref="xacml-context:Action"/>
    <xs:element ref="xacml-context:Environment" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Response" type="xacml-context:ResponseType"/>
<xs:complexType name="ResponseType">
  <xs:sequence>
    <xs:element ref="xacml-context:Result" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Subject" type="xacml-context:SubjectType"/>
<xs:complexType name="SubjectType">
  <xs:sequence>
    <xs:element ref="xacml-context:Attribute" minOccurs="0"
        maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="SubjectCategory" type="xs:anyURI" use="
      optional" default="urn:oasis:names:tc:xacml:1.0:subject-
      category:access-subject"/>
</xs:complexType>
<!-- -->
<xs:element name="Resource" type="xacml-context:ResourceType"/>
<xs:complexType name="ResourceType">
  <xs:sequence>
```

```
      <xs:element ref="xacml-context:ResourceContent" minOccurs="0"/>
      <xs:element ref="xacml-context:Attribute" minOccurs="0"
          maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- -->
  <xs:element name="ResourceContent" type="xacml-
      context:ResourceContentType"/>
  <xs:complexType name="ResourceContentType" mixed="true">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
          maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>
  <!-- -->
  <xs:element name="Action" type="xacml-context:ActionType"/>
  <xs:complexType name="ActionType">
    <xs:sequence>
      <xs:element ref="xacml-context:Attribute" minOccurs="0"
          maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- -->
  <xs:element name="Environment" type="xacml-context:EnvironmentType"/
      >
  <xs:complexType name="EnvironmentType">
    <xs:sequence>
      <xs:element ref="xacml-context:Attribute" minOccurs="0"
          maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- -->
  <xs:element name="Attribute" type="xacml-context:AttributeType"/>
  <xs:complexType name="AttributeType">
    <xs:sequence>
```

```
      <xs:element ref="xacml-context:AttributeValue"/>
    </xs:sequence>
    <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
    <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
    <xs:attribute name="Issuer" type="xs:string" use="optional"/>
    <xs:attribute name="IssueInstant" type="xs:dateTime" use="optional
        "/>
  </xs:complexType>
  <!-- -->
  <xs:element name="AttributeValue" type="xacml-
      context:AttributeValueType"/>
  <xs:complexType name="AttributeValueType" mixed="true">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
          maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>
  <!-- -->
  <xs:element name="Result" type="xacml-context:ResultType"/>
  <xs:complexType name="ResultType">
    <xs:sequence>
      <xs:element ref="xacml-context:Decision"/>
      <xs:element ref="xacml-context:Status"/>
      <xs:element ref="xacml:Obligations" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="ResourceId" type="xs:string" use="optional"/>
  </xs:complexType>
  <!-- -->
  <xs:element name="Decision" type="xacml-context:DecisionType"/>
  <xs:simpleType name="DecisionType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Permit"/>
      <xs:enumeration value="Deny"/>
      <xs:enumeration value="Indeterminate"/>
      <xs:enumeration value="NotApplicable"/>
```

```
    </xs:restriction>
  </xs:simpleType>
  <!-- -->
  <xs:element name="Status" type="xacml-context:StatusType"/>
  <xs:complexType name="StatusType">
    <xs:sequence>
      <xs:element ref="xacml-context:StatusCode"/>
      <xs:element ref="xacml-context:StatusMessage" minOccurs="0"/>
      <xs:element ref="xacml-context:StatusDetail" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!-- -->
  <xs:element name="StatusCode" type="xacml-context:StatusCodeType"/>
  <xs:complexType name="StatusCodeType">
    <xs:sequence>
      <xs:element ref="xacml-context:StatusCode" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Value" type="xs:anyURI" use="required"/>
  </xs:complexType>
  <!-- -->
  <xs:element name="StatusMessage" type="xs:string"/>
  <!-- -->
  <xs:element name="StatusDetail" type="xacml-context:StatusDetailType
      "/>
  <xs:complexType name="StatusDetailType">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
          maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

ANNEXE B

POLITIQUES D'EXEMPLE POUR LE PROFILE R-BAC POUR XACML

Listing B.1 – rps-professor.xml

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy" PolicySetId="
    RPS:professor:role" PolicyCombiningAlgId="&policy-combine;permit-
    overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="&function;string-equal">
          <AttributeValue DataType="&xml;string">professor</
              AttributeValue>
          <SubjectAttributeDesignator AttributeId="
              urn:fundp:attributes:role" DataType="&xml;string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <PolicySetIdReference>PPS:professor:role</PolicySetIdReference>
</PolicySet>
```

Listing B.2 – pps-professor.xml

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy" PolicySetId="
    PPS:professor:role" PolicyCombiningAlgId="&policy-combine;permit-
    overrides">
```

```
<Target>
  <Subjects><AnySubject/></Subjects>
  <Resources><AnyResource/></Resources>
  <Actions><AnyAction/></Actions>
</Target>

<Policy PolicyId="Permissions:specifically:for:the:professor:role"
    RuleCombiningAlgId="&rule-combine;permit-overrides">
  <Target>
    <Subjects><AnySubject/></Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>

  <Rule RuleId="Permission:to:create:webcampus-group"
      Effect="Permit">
    <Target>
      <Subjects><AnySubject/></Subjects>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="&function;string-match">
          <AttributeValue DataType="&xml;string">webcampus-group</
              AttributeValue>
          <ResourceAttributeDesignator AttributeId="&resource;resource
              -id" DataType="&xml;string"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="&function;string-match">
          <AttributeValue DataType="&xml;string">create</
              AttributeValue>
          <ActionAttributeDesignator AttributeId="&action;action-id"
              DataType="&xml;string"/>
          </ActionMatch>
```

```
        </Action>
      </Actions>
    </Target>
  </Rule>
</Policy>
```

Listing B.3 – rps-student.xml

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"
  PolicySetId="RPS:student:role" PolicyCombiningAlgId="&policy-combine
      ;permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="&function;string-equal">
          <AttributeValue DataType="&xml;string">student</
              AttributeValue>
          <SubjectAttributeDesignator AttributeId="
              urn:fundp:attributes:role"
            DataType="&xml;string" />
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <AnyResource />
    </Resources>
    <Actions>
      <AnyAction />
    </Actions>
  </Target>
  <PolicySetIdReference>PPS:student:role</PolicySetIdReference>
</PolicySet>
```

Listing B.4 – pps-student.xml

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy" PolicySetId="
    PPS:student:role" PolicyCombiningAlgId="&policy-combine;permit-
    overrides">
```

```
<Target>
  <Subjects><AnySubject/></Subjects>
  <Resources><AnyResource/></Resources>
  <Actions><AnyAction/></Actions>
</Target>

<Policy PolicyId="Permissions:specifically:for:the:student:role"
    RuleCombiningAlgId="&rule-combine;permit-overrides">
  <Target>
    <Subjects><AnySubject/></Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>

  <Rule RuleId="Permission:to:join:webcampus-group"
      Effect="Permit">
    <Target>
      <Subjects><AnySubject/></Subjects>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="&function;string-match">
          <AttributeValue DataType="&xml;string">webcampus-group</
              AttributeValue>
          <ResourceAttributeDesignator AttributeId="&resource;resource
              -id" DataType="&xml;string"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="&function;string-match">
          <AttributeValue DataType="&xml;string">join</AttributeValue>
          <ActionAttributeDesignator AttributeId="&action;action-id"
              DataType="&xml;string"/>
          </ActionMatch>
        </Action>
```

```
      </Actions>
    </Target>
  </Rule>
</Policy>
```

Listing B.5 – raps.xml

```
<Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy" PolicyId="
    Role:Assignment:Policy"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
      algorithm:permit-overrides">
  <Target>
    <Subjects>
      <AnySubject />
    </Subjects>
    <Resources>
      <AnyResource />
    </Resources>
    <Actions>
      <AnyAction />
    </Actions>
  </Target>

  <Rule RuleId="student:role:requirements" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                #string">Alice</AttributeValue>
            <SubjectAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0
                    :subject:subject-id" DataType="http://www.w3.org
                    /2001/XMLSchema#string" />
          </SubjectMatch>
        </Subject>
```

```
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                #string">Bob</AttributeValue>
            <SubjectAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0
                    :subject:subject-id" DataType="http://www.w3.org
                    /2001/XMLSchema#string" />
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                #string">student</AttributeValue>
            <ResourceAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0
                    :resource:resource-id" DataType="http://www.w3.org
                    /2001/XMLSchema#string" />
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                #string">enable</AttributeValue>
            <ActionAttributeDesignator AttributeId="
                urn:oasis:names:tc:xacml:1.0:action:action-id"
                DataType="http://www.w3.org/2001/XMLSchema#string" />
          </ActionMatch>
        </Action>
```

```
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-
          greater-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-
            one-and-only">
          <EnvironmentAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0
                  :environment:current-time" DataType="http://www.w3.org
                  /2001/XMLSchema#time" />
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
            time">08:00:00</AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-
          less-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-
            one-and-only">
          <EnvironmentAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0
                  :environment:current-time" DataType="http://www.w3.org
                  /2001/XMLSchema#time" />
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
            time">19:00:00</AttributeValue>
      </Apply>
    </Condition>
  </Rule>

  <Rule RuleId="professor:role:requirements" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
```

```
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                #string">Carole</AttributeValue>
            <SubjectAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0
                  :subject:subject-id" DataType="http://www.w3.org
                  /2001/XMLSchema#string" />
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                #string">professor</AttributeValue>
            <ResourceAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0
                  :resource:resource-id" DataType="http://www.w3.org
                  /2001/XMLSchema#string" />
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0
              :function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                #string">enable</AttributeValue>
            <ActionAttributeDesignator AttributeId="
                urn:oasis:names:tc:xacml:1.0:action:action-id"
              DataType="http://www.w3.org/2001/XMLSchema#string" />
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
```

```
</Policy>
```

Listing B.6 – alice-student-enable.xml

```
<Request>
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0
        :subject:subject-id"
            DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>Alice</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0
        :resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>student</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action
        -id"
            DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>enable</AttributeValue>
    </Attribute>
  </Action>
</Request>
```

Listing B.7 – bob-student-enable.xml

```
<Request>
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0
        :subject:subject-id"
            DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>Bob</AttributeValue>
    </Attribute>
  </Subject>
```

```
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0
        :resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>student</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action
        -id"
            DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>enable</AttributeValue>
    </Attribute>
  </Action>
</Request>
```

Listing B.8 – carole-professor-enable.xml

```
<Request>
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0
        :subject:subject-id"
            DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>Carole</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0
        :resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>professor</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action
        -id"
            DataType="http://www.w3.org/2001/XMLSchema#string">
```

```
        <AttributeValue>enable</AttributeValue>
      </Attribute>
    </Action>
20 </Request>
```

ANNEXE C

SCHEMA XML DU PROFILE CREATIVE COMMONS POUR ODRL

Listing C.1 – ODRL20-CC-profile-schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:o="http://w3.org/ns/odrl/2/" xmlns:xs="http://www.w3.
    org/2001/XMLSchema"
  targetNamespace="http://w3.org/ns/odrl/2/" elementFormDefault="
      qualified"
  attributeFormDefault="unqualified">

  <xs:element name="policy" type="o:policyCT" />
  <xs:complexType name="policyCT">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="o:permission" minOccurs="0" maxOccurs="
          unbounded" />
      <xs:element ref="o:prohibition" minOccurs="0" maxOccurs="
          unbounded" />
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="uid" type="o:URIQnameQcode" use="required" />
    <xs:attribute name="type" type="o:URIQnameQcode" use="required" />
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>

  <xs:element name="permission" type="o:permCT" />
  <xs:complexType name="permCT">
    <xs:sequence>
      <xs:element ref="o:asset" maxOccurs="unbounded" />
      <xs:element ref="o:action" />
      <xs:element ref="o:constraint" minOccurs="0" maxOccurs="
          unbounded" />
```

```
25      <xs:element ref="o:party" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="o:duty" minOccurs="0" maxOccurs="unbounded" />
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
          maxOccurs="unbounded" />
    </xs:sequence>
30  <xs:attributeGroup ref="o:idAttributes" />
  </xs:complexType>

  <xs:element name="prohibition" type="o:prohibCT" />
  <xs:complexType name="prohibCT">
35  <xs:sequence>
        <xs:element ref="o:asset" maxOccurs="unbounded" />
        <xs:element ref="o:action" />
        <xs:element ref="o:constraint" minOccurs="0" maxOccurs="
            unbounded" />
        <xs:element ref="o:party" minOccurs="0" maxOccurs="unbounded" />
40      <xs:any namespace="##other" processContents="lax" minOccurs="0"
          maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attributeGroup ref="o:idAttributes" />
  </xs:complexType>
45

  <xs:element name="duty" type="o:dutyCT" />
  <xs:complexType name="dutyCT">
    <xs:sequence>
        <xs:element ref="o:action" minOccurs="0" /> <!--Required Element
            -->
50      <xs:element ref="o:asset" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="o:constraint" minOccurs="0" maxOccurs="
            unbounded" />
        <xs:element ref="o:party" minOccurs="0" maxOccurs="unbounded" />
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
          maxOccurs="unbounded" />
55  </xs:sequence>
    <xs:attribute name="uid" type="o:URIQnameQcode" />
    <xs:attributeGroup ref="o:idAttributes" />
```

```
</xs:complexType>

<xs:element name="asset" type="o:assetCT" />
<xs:complexType name="assetCT">
  <xs:attribute name="uid" type="o:URIQnameQcode" /> <!--Required
      Attribute -->
  <xs:attribute name="relation" type="o:URIQnameQcode" /> <!--
      Required Attribute -->
  <xs:attributeGroup ref="o:idAttributes" />
</xs:complexType>

<xs:element name="party" type="o:partyCT" />
<xs:complexType name="partyCT">
  <xs:attribute name="uid" type="o:URIQnameQcode" /> <!--Required
      Attribute -->
  <xs:attribute name="function" type="o:URIQnameQcode" /> <!--
      Required Attribute -->
  <xs:attribute name="scope" type="o:URIQnameQcode" />
  <xs:attributeGroup ref="o:idAttributes" />
</xs:complexType>

<xs:element name="action" type="o:actionCT" />
<xs:complexType name="actionCT">
  <xs:attribute name="name" type="o:URIQnameQcode" /> <!--Required
      Attribute -->
  <xs:attributeGroup ref="o:idAttributes" />
</xs:complexType>

<xs:element name="constraint" type="o:constraintCT" />
<xs:complexType name="constraintCT">
  <xs:attribute name="name" type="o:URIQnameQcode" /> <!--Required
      Attribute -->
  <xs:attribute name="operator" type="o:URIQnameQcode" /> <!--
      Required Attribute -->
  <xs:attribute name="rightOperand" type="o:listOfValues" /> <!--
      Required Attribute -->
```

```
    <xs:attribute name="status" type="xs:string" />
    <xs:attributeGroup ref="o:idAttributes" />
  </xs:complexType>

  <xs:simpleType name="listOfValues">
    <xs:list itemType="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="URIQnameQcode">
    <xs:union memberTypes="xs:anyURI xs:QName o:QCode" />
  </xs:simpleType>

  <xs:simpleType name="QCode">
    <xs:restriction base="xs:string">
      <xs:pattern value="[^\s:]+:[^\s]+" />
    </xs:restriction>
  </xs:simpleType>

  <xs:attributeGroup name="idAttributes">
    <xs:attribute name="id" type="xs:ID" />
    <xs:attribute name="idref" type="xs:IDREF" />
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:attributeGroup>
</xs:schema>
```

ANNEXE D

FEUILLE XSL DE TRANSFORMATION DU PROFILE CREATIVE COMMONS POUR ODRL

Listing D.1 – odrl20-cc-to-XACML.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
    Transform">

  <xsl:output xmlns:xalan="http://xml.apache.org/xslt" method="xml"
      indent="yes" xalan:indent-amount="3" />

  <xsl:template match="policy">

    <xsl:choose>
      <xsl:when test="contains(@type,'request')">
        <xsl:element name="Request">
          <xsl:element name="Subject">
            <xsl:element name="Attribute">
              <xsl:attribute name="AttributeId">
                  urn:oasis:names:tc:xacml:1.0:subject:subject-id</
                  xsl:attribute>
              <xsl:attribute name="DataType">http://www.w3.org/2001/
                  XMLSchema#string</xsl:attribute>
              <xsl:element name="AttributeValue">
                <xsl:value-of select="permission[1]/party/@uid" />
              </xsl:element>
            </xsl:element>
          </xsl:element>
          <xsl:element name="Resource">
            <xsl:element name="Attribute">
              <xsl:attribute name="AttributeId">
                  urn:oasis:names:tc:xacml:1.0:resource:resource-id</
                  xsl:attribute>
```

```
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
            <xsl:element name="AttributeValue">
              <xsl:value-of select="permission[1]/asset/@uid" />
            </xsl:element>
          </xsl:element>
        </xsl:element>
        <xsl:element name="Action">
          <xsl:element name="Attribute">
            <xsl:attribute name="AttributeId">
                urn:oasis:names:tc:xacml:1.0:action:action-id</
                xsl:attribute>
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
            <xsl:element name="AttributeValue">
              <xsl:value-of select="permission[1]/action/@name" />
            </xsl:element>
          </xsl:element>
        </xsl:element>
      </xsl:element>
    </xsl:when>

    <xsl:when test="contains(@type,'offer')">
      <xsl:element name="Policy">
        <xsl:attribute name="PolicyId"><xsl:value-of
           select="@uid" /></xsl:attribute>
        <xsl:attribute name="RuleCombiningAlgId">
            urn:oasis:names:tc:xacml:1.0:rule-combining-
            algorithm:permit-overrides</xsl:attribute>

        <xsl:if test="local-name()='description'">
          <xsl:element name="Description">
            <xsl:apply-templates select="local-name()='description'"
                />
          </xsl:element>
        </xsl:if>
```

```
        <xsl:element name="Target">
          <xsl:apply-templates
            select="permission[1]/asset[@relation='http://w3.org/ns/
                odrl/vocab#target']" />
56      </xsl:element>
        <xsl:apply-templates select="permission" />
        <xsl:apply-templates select="prohibition" />

        <xsl:if test="permission/duty">
61        <xsl:element name="Obligations">
            <xsl:apply-templates select="permission/duty" />
          </xsl:element>
        </xsl:if>
      </xsl:element>
66  </xsl:when>

    <xsl:when test="contains(@type,'agreement')">
      <xsl:element name="Policy">
        <xsl:attribute name="PolicyId"><xsl:value-of
71        select="@uid" /></xsl:attribute>
        <xsl:attribute name="RuleCombiningAlgId">
            urn:oasis:names:tc:xacml:1.0:rule-combining-
            algorithm:permit-overrides</xsl:attribute>

        <xsl:if test="local-name()='description'">
          <xsl:element name="Description">
76          <xsl:apply-templates select="local-name()='description'"
                />
          </xsl:element>
        </xsl:if>

        <xsl:element name="Target">
81        <xsl:apply-templates
            select="permission[1]/party[@function='http://w3.org/ns/
                odrl/vocab#assignee']" />
```

```
            <xsl:apply-templates
              select="permission[1]/asset[@relation='http://w3.org/ns/
                odrl/vocab#target']" />
          </xsl:element>
86        <xsl:apply-templates select="permission" />
          <xsl:apply-templates select="prohibition" />

          <xsl:if test="permission/duty">
            <xsl:element name="Obligations">
91            <xsl:apply-templates select="permission/duty" />
            </xsl:element>
          </xsl:if>
        </xsl:element>
      </xsl:when>
96
      <xsl:otherwise>
      </xsl:otherwise>
    </xsl:choose>

101 </xsl:template>

    <xsl:template match="description">
      <xsl:apply-templates select="local-name()='valueString'" />
    </xsl:template>
106
    <xsl:template match="permission">
      <xsl:element name="Rule">
        <xsl:attribute name="RuleId">
          <xsl:value-of select="asset[@relation='http://w3.org/ns/odrl/
              vocab#target']/@uid" />:<xsl:value-of select="action/@name
              " />
111     </xsl:attribute>
        <xsl:attribute name="Effect">Permit</xsl:attribute>
        <xsl:element name="Target">
          <xsl:apply-templates select="action" />
        </xsl:element>
```

```
      <xsl:apply-templates select="constraint" />
    </xsl:element>
  </xsl:template>

  <xsl:template match="prohibition">
    <xsl:element name="Rule">
      <xsl:attribute name="RuleId">
        <xsl:value-of select="action/@name" />:<xsl:value-of select="
            asset[@relation='http://w3.org/ns/odrl/vocab#target']/@uid
            " />
      </xsl:attribute>
      <xsl:attribute name="Effect">Deny</xsl:attribute>
      <xsl:element name="Target">
        <xsl:apply-templates select="action" />
      </xsl:element>
      <xsl:apply-templates select="constraint" />
    </xsl:element>
  </xsl:template>

  <xsl:template match="asset">
    <xsl:element name="Resources">
      <xsl:element name="Resource">
        <xsl:element name="ResourceMatch">
          <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1.0
              :function:string-equal</xsl:attribute>
          <xsl:element name="AttributeValue">
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
            <xsl:value-of select="@uid" />
          </xsl:element>
          <xsl:element name="ResourceAttributeDesignator">
            <xsl:attribute name="AttributeId">
                urn:oasis:names:tc:xacml:1.0:resource:resource-id</
                xsl:attribute>
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
```

```
          </xsl:element>
146       </xsl:element>
      </xsl:element>
    </xsl:element>
  </xsl:template>

151 <xsl:template match="party">
    <xsl:element name="Subjects">
      <xsl:element name="Subject">
        <xsl:element name="SubjectMatch">
          <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1.0
              :function:string-equal</xsl:attribute>
156       <xsl:element name="AttributeValue">
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
            <xsl:value-of select="@uid" />
          </xsl:element>
          <xsl:element name="SubjectAttributeDesignator">
161         <xsl:attribute name="AttributeId">
                urn:oasis:names:tc:xacml:1.0:subject:subject-id</
                xsl:attribute>
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
          </xsl:element>
        </xsl:element>
      </xsl:element>
166   </xsl:element>
  </xsl:template>

  <xsl:template match="constraint">
    <xsl:choose>
171   <xsl:when test="contains(@name,'count') and @operator='http://w3
          .org/ns/odrl/vocab#eq'">
        <xsl:element name="Condition">
          <xsl:attribute name="FunctionId">urn:oasis:names:tc:xacml:1
              .0:function:integer-greater-than</xsl:attribute>
```

```
        <xsl:element name="AttributeValue">
          <xsl:attribute name="DataType">http://www.w3.org/2001/
              XMLSchema#integer</xsl:attribute>
176       <xsl:value-of select="@rightOperand"/>
        </xsl:element>
        <xsl:element name="AttributeValue">
          <xsl:attribute name="DataType">http://www.w3.org/2001/
              XMLSchema#integer</xsl:attribute>
          0
181     </xsl:element>
      </xsl:element>
    </xsl:when>
    <xsl:when test="contains(@name,'dateTime') and contains(
        @operator,'gteq')">
      <xsl:element name="Condition">
186     <xsl:attribute name="FunctionId">urn:oasis:names:tc:xacml:1
            .0:function:time-greater-than-or-equal</xsl:attribute>
        <xsl:element name="Apply">
          <xsl:attribute name="FunctionId">
              urn:oasis:names:tc:xacml:1.0:function:time-one-and-
              only</xsl:attribute>
          <xsl:element name="EnvironmentAttributeDesignator">
            <xsl:attribute name="AttributeId">
                urn:oasis:names:tc:xacml:1.0:environment:current-
                date</xsl:attribute>
191         <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#date</xsl:attribute>
          </xsl:element>
        </xsl:element>
        <xsl:element name="AttributeValue">
          <xsl:attribute name="DataType">http://www.w3.org/2001/
              XMLSchema#date</xsl:attribute>
196       <xsl:value-of select="@rightOperand"/>
        </xsl:element>
      </xsl:element>
    </xsl:when>
```

```
      <xsl:when test="contains(@name,'purpose') and @operator='http://
          w3.org/ns/odrl/vocab#eq'">
201     <xsl:element name="Condition">
          <xsl:attribute name="FunctionId">urn:oasis:names:tc:xacml:1
              .0:function:string-equal</xsl:attribute>
          <xsl:element name="Apply">
            <xsl:attribute name="FunctionId">
                urn:oasis:names:tc:xacml:1.0:function:string-one-and-
                only</xsl:attribute>
            <xsl:element name="EnvironmentAttributeDesignator">
206           <xsl:attribute name="AttributeId"><xsl:value-of select="
                  @name"/></xsl:attribute>
              <xsl:attribute name="DataType">http://www.w3.org/2001/
                  XMLSchema#string</xsl:attribute>
            </xsl:element>
          </xsl:element>
          <xsl:element name="AttributeValue">
211         <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
            <xsl:value-of select="@rightOperand"/>
          </xsl:element>
        </xsl:element>
      </xsl:when>
216   <xsl:when test="contains(@name,'spatial') and @operator='http://
          w3.org/ns/odrl/vocab#eq'">
        <xsl:element name="Condition">
          <xsl:attribute name="FunctionId">urn:oasis:names:tc:xacml:1
              .0:function:string-equal</xsl:attribute>
          <xsl:element name="Apply">
            <xsl:attribute name="FunctionId">
                urn:oasis:names:tc:xacml:1.0:function:string-one-and-
                only</xsl:attribute>
221         <xsl:element name="EnvironmentAttributeDesignator">
              <xsl:attribute name="AttributeId"><xsl:value-of select="
                  @name"/></xsl:attribute>
```

```
                <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#string</xsl:attribute>
              </xsl:element>
            </xsl:element>
            <xsl:element name="AttributeValue">
              <xsl:attribute name="DataType">http://www.w3.org/2001/
                  XMLSchema#string</xsl:attribute>
              <xsl:value-of select="@rightOperand"/>
            </xsl:element>
          </xsl:element>
        </xsl:when>
        <xsl:otherwise>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:template>

    <xsl:template match="duty">
        <xsl:element name="Obligation">
          <xsl:attribute name="ObligationId">
          <xsl:value-of select="../asset[@relation='http://w3.org/ns/
              odrl/vocab#target']/@uid"/>:<xsl:value-of select="../
              action/@name"/>:<xsl:value-of select="@uid"/>
          </xsl:attribute>
          <xsl:attribute name="FulfillOn">Permit</xsl:attribute>
          <xsl:element name="AttributeAssignment">
            <xsl:choose>
              <xsl:when test="action/@name='http://w3.org/ns/odrl/vocab#
                  pay'">
                <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#string</xsl:attribute>
                <xsl:attribute name="AttributeId">http://w3.org/ns/odrl/
                    vocab#pay</xsl:attribute>
                <xsl:value-of select="asset/@uid" />
              </xsl:when>
              <xsl:when test="action/@name='http://w3.org/ns/odrl/vocab#
                  attribute'">
```

```
251                <xsl:attribute name="DataType">http://www.w3.org/2001/
                       XMLSchema#string</xsl:attribute>
                   <xsl:attribute name="AttributeId">http://w3.org/ns/odrl/
                       vocab#attribute</xsl:attribute>
                   <xsl:value-of select="asset/@uid" />
                 </xsl:when>
                 <xsl:when test="action/@name">
256                <xsl:attribute name="DataType">http://www.w3.org/2001/
                       XMLSchema#string</xsl:attribute>
                   <xsl:attribute name="AttributeId"><xsl:value-of select="
                       action/@name" /></xsl:attribute>
                 </xsl:when>
                 <xsl:otherwise>
                   <xsl:attribute name="DataType">http://www.w3.org/2001/
                       XMLSchema#string</xsl:attribute>
261                <xsl:attribute name="AttributeId">see-ref-<xsl:value-of
                       select="@uid"/></xsl:attribute>
                 </xsl:otherwise>
               </xsl:choose>
             </xsl:element>
           </xsl:element>
266  </xsl:template>

     <xsl:template match="action">
       <xsl:choose>
         <xsl:when test="contains(@name,'commercialize')">
271        <xsl:element name="Actions">
             <xsl:element name="Action">
               <xsl:element name="ActionMatch">
                 <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1
                     .0:function:string-equal</xsl:attribute>
                 <xsl:element name="AttributeValue">
276                <xsl:attribute name="DataType">http://www.w3.org/2001/
                       XMLSchema#string</xsl:attribute>
                   http://w3.org/ns/odrl/vocab#commercialize
                 </xsl:element>
```

```
                <xsl:element name="ActionAttributeDesignator">
                  <xsl:attribute name="AttributeId">
                      urn:oasis:names:tc:xacml:1.0:action:action-id</
                      xsl:attribute>
                  <xsl:attribute name="DataType">http://www.w3.org/2001/
                      XMLSchema#string</xsl:attribute>
                </xsl:element>
              </xsl:element>
            </xsl:element>
          </xsl:element>
        </xsl:when>
        <xsl:when test="contains(@name,'copyReproduce')">
          <xsl:element name="Actions">
            <xsl:element name="Action">
              <xsl:element name="ActionMatch">
                <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1
                    .0:function:string-equal</xsl:attribute>
                <xsl:element name="AttributeValue">
                  <xsl:attribute name="DataType">http://www.w3.org/2001/
                      XMLSchema#string</xsl:attribute>
                  http://w3.org/ns/odrl/vocab#copyReproduce
                </xsl:element>
                <xsl:element name="ActionAttributeDesignator">
                  <xsl:attribute name="AttributeId">
                      urn:oasis:names:tc:xacml:1.0:action:action-id</
                      xsl:attribute>
                  <xsl:attribute name="DataType">http://www.w3.org/2001/
                      XMLSchema#string</xsl:attribute>
                </xsl:element>
              </xsl:element>
            </xsl:element>
          </xsl:element>
        </xsl:when>
        <xsl:when test="contains(@name,'distribute')">
          <xsl:element name="Actions">
            <xsl:element name="Action">
```

```
            <xsl:element name="ActionMatch">
              <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1
                  .0:function:string-equal</xsl:attribute>
              <xsl:element name="AttributeValue">
                <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#string</xsl:attribute>
                http://w3.org/ns/odrl/vocab#distribute
              </xsl:element>
              <xsl:element name="ActionAttributeDesignator">
                <xsl:attribute name="AttributeId">
                    urn:oasis:names:tc:xacml:1.0:action:action-id</
                    xsl:attribute>
                <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#string</xsl:attribute>
              </xsl:element>
            </xsl:element>
          </xsl:element>
        </xsl:element>
      </xsl:when>
      <xsl:when test="contains(@name,'deriveModify')">
        <xsl:element name="Actions">
          <xsl:element name="Action">
            <xsl:element name="ActionMatch">
              <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1
                  .0:function:string-equal</xsl:attribute>
              <xsl:element name="AttributeValue">
                <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#string</xsl:attribute>
                http://w3.org/ns/odrl/vocab#deriveModify
              </xsl:element>
              <xsl:element name="ActionAttributeDesignator">
                <xsl:attribute name="AttributeId">
                    urn:oasis:names:tc:xacml:1.0:action:action-id</
                    xsl:attribute>
                <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#string</xsl:attribute>
```

```
            </xsl:element>
          </xsl:element>
        </xsl:element>
336     </xsl:element>
      </xsl:when>
      <xsl:when test="contains(@name,'share')">
        <xsl:element name="Actions">
          <xsl:element name="Action">
341         <xsl:element name="ActionMatch">
              <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1
                  .0:function:string-equal</xsl:attribute>
              <xsl:element name="AttributeValue">
                <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#string</xsl:attribute>
                http://w3.org/ns/odrl/vocab#share
346           </xsl:element>
              <xsl:element name="ActionAttributeDesignator">
                <xsl:attribute name="AttributeId">
                    urn:oasis:names:tc:xacml:1.0:action:action-id</
                    xsl:attribute>
                <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#string</xsl:attribute>
              </xsl:element>
351         </xsl:element>
          </xsl:element>
        </xsl:element>
      </xsl:when>
      <xsl:when test="contains(@name,'highIncomeNationUse')">
356     <xsl:element name="Actions">
          <xsl:element name="Action">
            <xsl:element name="ActionMatch">
              <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1
                  .0:function:string-equal</xsl:attribute>
              <xsl:element name="AttributeValue">
361             <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#string</xsl:attribute>
```

```
                http://creativecommons.org/ns#highIncomeNationUse
            </xsl:element>
            <xsl:element name="ActionAttributeDesignator">
              <xsl:attribute name="AttributeId">
                  urn:oasis:names:tc:xacml:1.0:action:action-id</
                  xsl:attribute>
              <xsl:attribute name="DataType">http://www.w3.org/2001/
                  XMLSchema#string</xsl:attribute>
            </xsl:element>
          </xsl:element>
        </xsl:element>
      </xsl:element>
    </xsl:when>
    <xsl:when test="contains(@name,'display')">
      <xsl:element name="Actions">
        <xsl:element name="Action">
          <xsl:element name="ActionMatch">
            <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1
                .0:function:string-equal</xsl:attribute>
            <xsl:element name="AttributeValue">
              <xsl:attribute name="DataType">http://www.w3.org/2001/
                  XMLSchema#string</xsl:attribute>
              display
            </xsl:element>
            <xsl:element name="ActionAttributeDesignator">
              <xsl:attribute name="AttributeId">
                  urn:oasis:names:tc:xacml:1.0:action:action-id</
                  xsl:attribute>
              <xsl:attribute name="DataType">http://www.w3.org/2001/
                  XMLSchema#string</xsl:attribute>
            </xsl:element>
          </xsl:element>
        </xsl:element>
      </xsl:element>
    </xsl:when>
    <xsl:when test="contains(@name,'print')">
```

```
        <xsl:element name="Actions">
391       <xsl:element name="Action">
            <xsl:element name="ActionMatch">
              <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1
                  .0:function:string-equal</xsl:attribute>
              <xsl:element name="AttributeValue">
                <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#string</xsl:attribute>
396             print
              </xsl:element>
              <xsl:element name="ActionAttributeDesignator">
                <xsl:attribute name="AttributeId">
                    urn:oasis:names:tc:xacml:1.0:action:action-id</
                    xsl:attribute>
                <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#string</xsl:attribute>
401           </xsl:element>
            </xsl:element>
          </xsl:element>
        </xsl:element>
      </xsl:when>
406   <xsl:otherwise>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

411 </xsl:stylesheet>
```

ANNEXE E

POLITIQUES D'EXEMPLE DU PROFILE CREATIVE COMMONS POUR ODRL

Listing E.1 – cc-odrl20-agreement-music.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<policy xmlns="http://w3.org/ns/odrl/2/"
  type="http://w3.org/ns/odrl/vocab#agreement" uid="
      urn:anita:volcanolove:01">
  <dcds:description xmlns:dcds="http://purl.org/dc/xmlns/2008/09/01/dc
      -ds-xml/">
    <dcds:statement dcds:propertyURI="http://purl.org/dc/terms/source"
                    dcds:valueURI="http://creativecommons.org/licenses/by-
                        nc-sa/2.0">
    <dcds:valueString>Attribution-NonCommercial-ShareAlike 2.0 Generic
        (CC BY-NC-SA 2.0)</dcds:valueString>
    </dcds:statement>
  </dcds:description>
  <permission>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3" relation="http:
        //w3.org/ns/odrl/vocab#target"/>
    <action name="http://w3.org/ns/odrl/vocab#copyReproduce"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org
        /ns/odrl/vocab#assigner"/>
    <party id="p2" uid="urn:fundp:bob" function="http://w3.org/ns/odrl
        /vocab#assignee"/>
    <duty uid="d1">
          <action name="http://w3.org/ns/odrl/vocab#attachPolicy"/>
        </duty>
    <duty uid="d2">
          <action name="http://w3.org/ns/odrl/vocab#shareAlike"/>
        </duty>
    <duty uid="d3">
          <action name="http://w3.org/ns/odrl/vocab#attribute"/>
```

```
        </duty>
  </permission>
  <permission>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3" relation="http:
        //w3.org/ns/odrl/vocab#target"/>
    <action name="http://w3.org/ns/odrl/vocab#distribute"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org
        /ns/odrl/vocab#assigner"/>
    <party id="p2"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </permission>
  <permission>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3" relation="http:
        //w3.org/ns/odrl/vocab#target"/>
    <action name="http://w3.org/ns/odrl/vocab#deriveModify"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org
        /ns/odrl/vocab#assigner"/>
    <party id="p2"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </permission>
  <prohibition>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3" relation="http:
        //w3.org/ns/odrl/vocab#target"/>
    <action name="http://w3.org/ns/odrl/vocab#commercialize"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org
        /ns/odrl/vocab#assigner"/>
    <party id="p2"/>
  </prohibition>
</policy>
```

Listing E.2 – cc-odrl20-agreement-lyrics.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<policy xmlns="http://w3.org/ns/odrl/2/"
  type="http://w3.org/ns/odrl/vocab#agreement"
  uid="urn:anita:lyrics:volcanolove:01">
  <permission>
    <asset uid="urn:anita-music:lyrics:volcanoLove.pdf" relation="
        http://w3.org/ns/odrl/vocab#target"/>
    <action name="http://w3.org/ns/odrl/vocab#distribute"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org
        /ns/odrl/vocab#assigner"/>
    <party id="p2" uid="urn:fundp:alice" function="http://w3.org/ns/
        odrl/vocab#assignee"/>
    <duty uid="d1">
          <action name="http://w3.org/ns/odrl/vocab#attachPolicy"/>
        </duty>
    <duty uid="d2">
          <action name="http://w3.org/ns/odrl/vocab#shareAlike"/>
        </duty>
    <duty uid="d3">
          <action name="http://w3.org/ns/odrl/vocab#attribute"/>
        </duty>
  </permission>
  <permission>
    <asset uid="urn:anita-music:lyrics:volcanoLove.pdf" relation="
        http://w3.org/ns/odrl/vocab#target"/>
    <action name="http://w3.org/ns/odrl/vocab#display"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org
        /ns/odrl/vocab#assigner"/>
    <party id="p2">
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </permission>
  <permission>
    <asset uid="urn:anita-music:lyrics:volcanoLove.pdf" relation="
        http://w3.org/ns/odrl/vocab#target"/>
    <action name="http://w3.org/ns/odrl/vocab#print"/>
```

```
    <constraint name="http://w3.org/ns/odrl/vocab#count" operator="
        http://w3.org/ns/odrl/vocab#eq" rightOperand="1"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org
        /ns/odrl/vocab#assigner"/>
    <party id="p2">
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </permission>
  <prohibition>
    <asset uid="urn:anita-music:lyrics:volcanoLove.pdf" relation="
        http://w3.org/ns/odrl/vocab#target"/>
    <action name="http://w3.org/ns/odrl/vocab#commercialize"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org
        /ns/odrl/vocab#assigner"/>
    <party id="p2">
  </prohibition>
</policy>
```

Listing E.3 – bob-music-copyReproduce.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<policy xmlns="http://w3.org/ns/odrl/2/" type="http://w3.org/ns/odrl/
    vocab#request" uid="urn:request:bob:music:copyReproduce">
    <permission>
        <asset uid="urn:anita-music:songs:volcanoLove.mp3" relation="
            http://w3.org/ns/odrl/vocab#target"/>
        <action name="http://w3.org/ns/odrl/vocab#copyReproduce"/>
        <party uid="urn:fundp:bob" function="http://w3.org/ns/odrl/
            vocab#assignee"/>
    </permission>
</policy>
```

Listing E.4 – alice-lyrics-print.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<policy xmlns="http://w3.org/ns/odrl/2/" type="http://w3.org/ns/odrl/
    vocab#request" uid="urn:request:alice:lyrics:print">
```

```
    <permission>
        <asset uid="urn:anita-music:lyrics:volcanoLove.pdf" relation="
            http://w3.org/ns/odrl/vocab#target"/>
        <action name="http://w3.org/ns/odrl/vocab#print"/>
        <party uid="urn:fundp:alice" function="http://w3.org/ns/odrl/
            vocab#assignee"/>
    </permission>
</policy>
```

ANNEXE F

SCHEMA XML DU PROFILE MOBILE POUR ISO REL

Listing F.1 – iso-rel-mobile-profile.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:mpeg:mpeg21:2003:01-REL-R-NS"
    xmlns:mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS" xmlns:sx="
    urn:mpeg:mpeg21:2003:01-REL-SX-NS" xmlns:dsig="http://www.w3.org
    /2000/09/xmldsig#" xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:r="
    urn:mpeg:mpeg21:2003:01-REL-R-NS" xmlns:sccns="urn:uddi-
    org:schemaCentricC14N:2002-07-10" elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
      schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xsd:import namespace="http://www.w3.org/2001/04/xmlenc#"
      schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core
      -20021210/xenc-schema.xsd"/>
  <xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
      schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core
      -20020212/xmldsig-core-schema.xsd"/>
  <xsd:import namespace="urn:uddi-org:schemaCentricC14N:2002-07-10"
      schemaLocation="http://www.uddi.org/schema/
      SchemaCentricCanonicalization.xsd"/>
<!-- Core part -->
  <!-- -->
  <!-- Elements -->
  <!-- -->
  <xsd:element name="allConditions" type="r:AllConditions"
      substitutionGroup="r:condition">
    <xsd:annotation>
      <xsd:documentation>A container of other conditions, all of which
          must be met simultaneously.</xsd:documentation>
    </xsd:annotation>
```

```
</xsd:element>
<xsd:element name="condition" type="r:Condition" substitutionGroup="
    r:licensePart">
  <xsd:annotation>
    <xsd:documentation>Specifies the terms, conditions, and
        obligations under which rights can be exercised. The
        conditions in a grant can have a known structure specifying
        that they all must be met simultaneously (a conjunction) or
        only one of them must be met. This known structure enables a
         generic engine with no specific knowledge of the semantics
        of the rights or conditions to compute the grants in effect
        through a chain of delegated rights.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="datum" type="r:Datum">
  <xsd:annotation>
    <xsd:documentation>Defines one raw parameter to be passed to the
         service.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="grant" type="r:Grant" substitutionGroup="
    r:resource">
  <xsd:annotation>
    <xsd:documentation>The quantum within the license that bestows
        an authorization upon some principal. It conveys to a
        particular principal the sanction to exercise an identified
        right against an identified resource, possibly subject to
        first fulfilling some conditions.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="grantGroup" type="r:GrantGroup" substitutionGroup
    ="r:resource">
  <xsd:annotation>
    <xsd:documentation>A container of several grants. A grantGroup
        does not define any semantic association, ordering
        relationship, and so on, between the grants it contains.</
```

```
        xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="issuer" type="r:Issuer">
    <xsd:annotation>
      <xsd:documentation>Identifies the entity who signs the license,
          attesting to its validity. If more than one issuer signs the
           license, it is as if each signed it independently; one
          license with several issuers is equivalent to several copies
           of the same license, each with one issuer. Indeed, such a
          syntactic transformation can feasibly be made while
          preserving the signature validity.</xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="keyHolder" type="r:KeyHolder" substitutionGroup="
      r:principal">
    <xsd:annotation>
      <xsd:documentation>Identifies a principal who possesses a
          particular key. Typically, the key is a private key
          corresponding to a public key identified by this element,
          but it may be a symmetric key. The public key can be
          identified by several mechanisms defined in the XML Digital
          Signature specification. </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="license" type="r:License">
    <xsd:annotation>
      <xsd:documentation>A container of one or more grants, each of
          which conveys to a principal a right to a resource under
          certain conditions. The license also specifies its issuer
          and other administrative information.</xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="licenseGroup" type="r:LicenseGroup">
    <xsd:annotation>
```

```
      <xsd:documentation>A container of licenses. A licenseGroup does
          not define any semantic association, ordering relationship,
          and so on, between the licenses it contains.</
          xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="licensePart" type="r:LicensePart">
    <xsd:annotation>
      <xsd:documentation>An abstract element from which the various
          specific parts of a license are derived. This element
          defines attributes common to all parts of a license.</
          xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="principal" type="r:Principal" substitutionGroup="
      r:resource">
    <xsd:annotation>
      <xsd:documentation>Represents the unique identification of a
          party involved in granting or exercising rights. Each
          principal identifies exactly one party.</xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="resource" type="r:Resource" substitutionGroup="
      r:licensePart">
    <xsd:annotation>
      <xsd:documentation>The "noun" to which a principal can be
          granted a right. A resource can be a digital work (such as
          an e-book, an audio or video file, or an image), a service (
          such as an email service or B2B transaction service), or
          even a piece of information that can be owned by a principal
           (such as a name or an email address). </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="right" type="r:Right" substitutionGroup="
      r:licensePart">
    <xsd:annotation>
```

```
      <xsd:documentation>The "verb" that a principal can be granted to
          exercise against some resource under some condition.
          Typically, a right specifies an action (or activity) or a
          class of actions that a principal may perform on or using
          the associated resource. </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="validityInterval" type="r:ValidityInterval"
      substitutionGroup="r:condition">
    <xsd:annotation>
      <xsd:documentation>Identifies the time interval during which the
          associated right is valid.</xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="serviceDescription" type="r:ServiceDescription">
    <xsd:annotation>
      <xsd:documentation>Identifies the metadata and location of a
          service.</xsd:documentation>
    </xsd:annotation>
  </xsd:element>
    <xsd:element name="serviceReference" type="r:ServiceReference"
        substitutionGroup="r:resource">
    <xsd:annotation>
      <xsd:documentation>Provides the means to locate and interact
          with a concrete service. Specifically, this element
          identifies both an endpoint/address at which the service is
          located and meta information by which the type or interface
          for the endpoint can be understood.</xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <!-- -->
  <!--Complex Types-->
  <!-- -->
  <xsd:complexType name="Datum">
    <xsd:annotation>
```

```
      <xsd:documentation>Defines one raw parameter to be passed to the
          service.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:LicensePart">
        <xsd:sequence minOccurs="0">
          <xsd:any namespace="##any" processContents="lax"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="AllConditions">
    <xsd:annotation>
      <xsd:documentation>A container of other conditions, all of which
          must be met simultaneously.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:Condition">
        <xsd:sequence>
          <xsd:element ref="r:condition" minOccurs="0" maxOccurs="
              unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
    <xsd:complexType name="Condition">
    <xsd:annotation>
      <xsd:documentation>Specifies the terms, conditions, and
          obligations under which rights can be exercised. The
          conditions in a grant can have a known structure specifying
          that they all must be met simultaneously (a conjunction) or
          only one of them must be met. This known structure enables a
           generic engine with no specific knowledge of the semantics
          of the rights or conditions to compute the grants in effect
          through a chain of delegated rights.</xsd:documentation>
    </xsd:annotation>
```

```
    <xsd:complexContent>
      <xsd:extension base="r:LicensePart"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Grant">
    <xsd:annotation>
      <xsd:documentation>The quantum within the license that bestows
          an authorization upon some principal. It conveys to a
          particular principal the sanction to exercise an identified
          right against an identified resource, possibly subject to
          first fulfilling some conditions.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:Resource">
          <xsd:sequence>
            <xsd:element ref="r:keyHolder" minOccurs="0"/>
            <xsd:element ref="r:right"/>
            <xsd:element ref="r:diReference"/>
            <xsd:element ref="r:condition" minOccurs="0"/>
          </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GrantGroup">
    <xsd:annotation>
      <xsd:documentation>A container of several grants. A grantGroup
          does not define any semantic association, ordering
          relationship, and so on, between the grants it contains.</
          xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:Resource">
          <xsd:sequence>
            <xsd:choice maxOccurs="unbounded">
              <xsd:element ref="r:grant"/>
              <xsd:element ref="r:grantGroup"/>
```

```
            </xsd:choice>
152         </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Issuer">
157   <xsd:annotation>
      <xsd:documentation>Describes information associated with each
          issuer (signer) of a license. The SignedInfo in the
          Signature must contain a Reference that covers the whole
          license except for its immediate issuer children. Optionally
          , the SignedInfo may contain a second Reference that covers
          the details of the specific issuer. Boilerplate XPATH
          Transforms can be used for each Reference.</
          xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element ref="r:keyHolder"/>
162   <xsd:element name="details" type="r:IssuerDetails" minOccurs="0"
          >
      <xsd:annotation>
        <xsd:documentation>Issuer-specific contributions to the
            license.</xsd:documentation>
      </xsd:annotation>
      </xsd:element>
167   </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="IssuerDetails">
    <xsd:annotation>
      <xsd:documentation>Issuer-specific contributions to the license.
          </xsd:documentation>
172   </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="timeOfIssue" type="xsd:dateTime" minOccurs="0
          ">
        <xsd:annotation>
```

```
          <xsd:documentation>The date at which the license was issued,
               as attested to by this issuer. For many purposes,
              validators cannot rely on this assertion, but instead
              require some disinterested third part to attest to the
              date of issuance.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="KeyHolder">
    <xsd:annotation>
      <xsd:documentation>Identifies a principal who possesses a
          particular key. Typically, the key is a private key
          corresponding to a public key identified by this element,
          but it may be a symmetric key. The public key can be
          identified by several mechanisms defined in the XML Digital
          Signature specification. </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:Principal">
        <xsd:sequence>
          <xsd:element name="info" type="dsig:KeyInfoType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="License">
    <xsd:annotation>
      <xsd:documentation>A container of one or more grants, each of
          which conveys to a principal a right to a resource under
          certain conditions. The license also specifies its issuer
          and other administrative information. The optional licenseID
           attribute uniquely and globally identify this license over
          space and time. Note (by way of comparison to validity
          intervals in, say, X509) that as a pragmatic matter, each
          right in a license usually contains a time condition to
```

```
        limit its validity time.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:choice minOccurs="1" maxOccurs="unbounded">
        <xsd:element ref="r:grant"/>
        <xsd:element ref="r:grantGroup"/>
      </xsd:choice>
      <xsd:element ref="r:issuer"/>
    </xsd:sequence>
    <xsd:attribute name="profileCompliance" type="r:ProfileCompliance"
        />
  </xsd:complexType>
  <xsd:complexType name="LicenseGroup">
    <xsd:annotation>
      <xsd:documentation>A container of licenses. A licenseGroup does
          not define any semantic association, ordering relationship,
          and so on, between the licenses it contains.</
          xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element ref="r:license" minOccurs="0" maxOccurs="unbounded"
          />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="LicensePart">
    <xsd:annotation>
      <xsd:documentation>An abstract element from which the various
          specific parts of a license are derived. This element
          defines attributes common to all parts of a license. A
          license part can have an identifier or reference an
          identifier defined elsewhere in this license. This mechanism
           reduces verbosity by defining commonly-used elements in one
           place and referencing them elsewhere. However, this is a
          purely syntactic shorthand; no semantic connection between
          the definition site and use site is implied. </
          xsd:documentation>
```

```
    </xsd:annotation>
    <xsd:attribute name="licensePartId" type="r:LicensePartId" use="
        optional"/>
    <xsd:attribute name="licensePartIdRef" type="r:LicensePartId" use=
        "optional"/>
    <xsd:attribute name="varRef" type="r:VariableName" use="optional"/
        >
  </xsd:complexType>
  <xsd:complexType name="Principal">
    <xsd:annotation>
      <xsd:documentation>Represents the unique identification of a
          party involved in granting or exercising rights. Each
          principal identifies exactly one party.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:Resource"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Resource">
    <xsd:annotation>
      <xsd:documentation>The "noun" to which a principal can be
          granted a right. A resource can be a digital work (such as
          an e-book, an audio or video file, or an image), a service (
          such as an email service or B2B transaction service), or
          even a piece of information that can be owned by a principal
           (such as a name or an email address). </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:LicensePart"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Right">
    <xsd:annotation>
      <xsd:documentation>The "verb" that a principal can be granted to
           exercise against some resource under some condition.
          Typically, a right specifies an action (or activity) or a
```

```
        class of actions that a principal may perform on or using
        the associated resource. </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:LicensePart"/>
    </xsd:complexContent>
  </xsd:complexType>
    <xsd:complexType name="ServiceDescription">
    <xsd:annotation>
      <xsd:documentation>Identifies the metadata and location of a
          service.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:LicensePart"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ServiceReference">
    <xsd:annotation>
      <xsd:documentation>Provides the means to locate and interact
          with a concrete service. Specifically, this element
          identifies both an endpoint/address at which the service is
          located and meta information by which the type or interface
          for the endpoint can be understood.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:Resource">
        <xsd:sequence minOccurs="0">
          <xsd:element ref="r:serviceDescription"/>
          <xsd:element name="serviceParameters" minOccurs="0">
            <xsd:annotation>
              <xsd:documentation>Provides contextual parameters that
                  may be needed to interact with the service. The
                  exact interpretation of each parameter is specific
                  to the semantics of the service and is not specified
                   here.</xsd:documentation>
            </xsd:annotation>
```

```
            <xsd:complexType>
267           <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                <xsd:element ref="r:datum"/>
                <xsd:element name="transforms" type="
                    dsig:TransformsType" minOccurs="0">
                  <xsd:annotation>
                    <xsd:documentation>Lists optional transformations
                        to be applied in sequence over datum.</
                        xsd:documentation>
272               </xsd:annotation>
                </xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
277       </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ValidityInterval">
282   <xsd:annotation>
      <xsd:documentation>Identifies the time interval during which the
          associated right is valid.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:Condition">
287     <xsd:sequence>
          <xsd:element name="notBefore" type="xsd:dateTime" minOccurs=
              "0">
            <xsd:annotation>
              <xsd:documentation>Identifies the beginning of the
                  interval.</xsd:documentation>
            </xsd:annotation>
292       </xsd:element>
          <xsd:element name="notAfter" type="xsd:dateTime" minOccurs="
              0">
            <xsd:annotation>
```

```
                <xsd:documentation>Identifies the end of the interval.</
                    xsd:documentation>
            </xsd:annotation>
297         </xsd:element>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
302 <!-- -->
  <!-- Simple Types-->
  <!-- -->
  <xsd:simpleType name="LicensePartId">
    <xsd:annotation>
307   <xsd:documentation>Identifier for a license part. Using this
          identifier, commonly-used elements can be defined in one
          place and referenced elsewhere, thus reducing verbosity.</
          xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:NCName"/>
  </xsd:simpleType>
  <xsd:simpleType name="VariableName">
312 <xsd:annotation>
      <xsd:documentation>The name of a variable.</xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:NCName"/>
  </xsd:simpleType>
317
<!-- Standard extension part -->
  <!-- -->
  <!-- Elements -->
  <!-- -->
322 <xsd:element name="exerciseLimit" type="r:ExerciseLimit"
      substitutionGroup="r:condition">
    <xsd:annotation>
      <xsd:documentation>Indicates a maximum number of times that the
          right may be exercised.</xsd:documentation>
```

```
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="stateDistinguisher" block="#all"
      substitutionGroup="r:licensePart" final="#all">
    <xsd:complexType mixed="true">
      <xsd:complexContent mixed="true">
        <xsd:extension base="r:LicensePart"/>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="validityIntervalFloating" type="
      r:ValidityIntervalFloating" substitutionGroup="r:condition">
    <xsd:annotation>
      <xsd:documentation>Represents an interval of time that begins
          with the first exercise of a right.</xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <!-- -->
  <!-- Complex Types -->
  <!-- -->
  <xsd:complexType name="ExerciseLimit">
    <xsd:annotation>
      <xsd:documentation>Indicates a maximum number of times that the
          right may be exercised.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:StatefulCondition">
        <xsd:sequence>
          <xsd:element name="count" type="xsd:integer"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="StatefulCondition">
    <xsd:annotation>
```

```
      <xsd:documentation>A condition that requires that some state be
          referenced or manipulated to be satisfied.</
          xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:Condition">
        <xsd:sequence minOccurs="0">
          <xsd:element ref="r:serviceReference"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ValidityIntervalFloating">
    <xsd:annotation>
      <xsd:documentation>Represents an interval of time that begins
          with the first exercise of a right.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="r:StatefulCondition">
        <xsd:sequence>
          <xsd:element name="duration" type="xsd:duration"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Simple Types -->
  <xsd:simpleType name="ProfileCompliance">
    <xsd:list itemType="xsd:QName"/>
  </xsd:simpleType>
  <!-- Attributes -->
  <xsd:attribute name="profileCompliance" type="r:ProfileCompliance"/>

<!--  Multimedia extension part -->
  <!-- -->
  <!-- Rights -->
  <!-- -->
```

```
  <xsd:element name="play" type="r:Play" substitutionGroup="r:right"/>
  <xsd:element name="print" type="r:Print" substitutionGroup="r:right"
      />
  <xsd:element name="execute" type="r:Execute" substitutionGroup="
      r:right"/>
  <xsd:complexType name="Play">
    <xsd:complexContent>
      <xsd:extension base="r:Right"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Print">
    <xsd:complexContent>
      <xsd:extension base="r:Right"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Execute">
    <xsd:complexContent>
      <xsd:extension base="r:Right"/>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- -->
  <!-- Digital Item Resource -->
  <!-- -->
  <xsd:complexType name="DiReference">
    <xsd:complexContent>
      <xsd:extension base="r:Resource">
        <xsd:sequence>
          <xsd:element name="identifier" type="xsd:anyURI"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="diReference" type="r:DiReference"
      substitutionGroup="r:resource"/>

<!-- -->
```

```
422 <!-- XML Signature Syntax and Processing part -->
    <element name="KeyName" type="string"/>
    </xsd:schema>
```

ANNEXE G

FEUILLE XSL DE TRANSFORMATION DU PROFILE MOBILE POUR ISO REL

Listing G.1 – iso-rel-mobile-profile-to-XACML.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
    Transform"
  xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
  xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xalan="http://xml.apache.org/xslt">

<xsl:output method="xml" indent="yes" xalan:indent-amount="3" />

  <xsl:template match="license">
    <xsl:choose>
    <xsl:when test="starts-with(@licenseId, 'request')">
      <xsl:element name="Request">
        <xsl:element name="Subject">
          <xsl:element name="Attribute">
            <xsl:attribute name="AttributeId">
                urn:oasis:names:tc:xacml:1.0:subject:subject-id</
                xsl:attribute>
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
            <xsl:element name="AttributeValue">
              <xsl:value-of select="grant[1]/party/@uid" />
            </xsl:element>
          </xsl:element>
        </xsl:element>
        <xsl:element name="Resource">
          <xsl:element name="Attribute">
            <xsl:attribute name="AttributeId">
                urn:oasis:names:tc:xacml:1.0:resource:resource-id</
```

```
            xsl:attribute>
        <xsl:attribute name="DataType">http://www.w3.org/2001/
            XMLSchema#string</xsl:attribute>
        <xsl:element name="AttributeValue">
          <xsl:value-of select="grant[1]/diReference/identifier" /
              >
        </xsl:element>
      </xsl:element>
    </xsl:element>
    <xsl:element name="Action">
      <xsl:element name="Attribute">
        <xsl:attribute name="AttributeId">
            urn:oasis:names:tc:xacml:1.0:action:action-id</
            xsl:attribute>
        <xsl:attribute name="DataType">http://www.w3.org/2001/
            XMLSchema#string</xsl:attribute>
        <xsl:element name="AttributeValue">
          <xsl:choose>
            <xsl:when test="grant[1]/play">
              play
            </xsl:when>
            <xsl:when test="grant[1]/print">
              print
            </xsl:when>
            <xsl:when test="grant[1]/execute">
              execute
            </xsl:when>
            <xsl:otherwise>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:element>
      </xsl:element>
    </xsl:element>
  </xsl:element>
</xsl:when>
```

```
  <xsl:otherwise>
    <xsl:element name="Policy">
      <xsl:attribute name="PolicyId">
        <xsl:value-of select="grant[1]/diReference/identifier"/><
            xsl:text>:</xsl:text><xsl:value-of select="grant[1]/
            keyHolder/info/KeyName"/>
      </xsl:attribute>
      <xsl:attribute name="RuleCombiningAlgId">
          urn:oasis:names:tc:xacml:1.0:rule-combining-
          algorithm:permit-overrides</xsl:attribute>

      <xsl:element name="Description">
        issued by : <xsl:apply-templates select="issuer/keyHolder/
            info/KeyName"/>
      </xsl:element>

      <xsl:element name="Target">
        <xsl:apply-templates select="grant[1]/keyHolder"/>
        <xsl:apply-templates select="grant[1]/diReference"/>
      </xsl:element>

      <xsl:apply-templates select="grant"/>
    </xsl:element>
  </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<xsl:template match="grant">
  <xsl:element name="Rule">
    <xsl:attribute name="RuleId">
      <xsl:text>rule</xsl:text><xsl:value-of select="count(preceding
          -sibling::grant) + 1" /> <!-- "count(self::*/preceding-
          sibling::*) + 1" -->
    </xsl:attribute>
    <xsl:attribute name="Effect">Permit</xsl:attribute>
    <xsl:element name="Target">
```

```
        <xsl:apply-templates select="play" />
        <xsl:apply-templates select="print" />
        <xsl:apply-templates select="execute" />
      </xsl:element>
      <xsl:apply-templates select="validityInterval" />
      <xsl:apply-templates select="validityIntervalFloating" />
      <xsl:apply-templates select="exerciseLimit" />
      <xsl:apply-templates select="allConditions" />
    </xsl:element>
  </xsl:template>

  <xsl:template match="diReference">
    <xsl:element name="Resources">
      <xsl:element name="Resource">
        <xsl:element name="ResourceMatch">
          <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1.0
              :function:string-equal</xsl:attribute>
          <xsl:element name="AttributeValue">
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
            <xsl:value-of select="identifier" />
          </xsl:element>
          <xsl:element name="ResourceAttributeDesignator">
            <xsl:attribute name="AttributeId">
                urn:oasis:names:tc:xacml:1.0:resource:resource-id</
                xsl:attribute>
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
          </xsl:element>
        </xsl:element>
      </xsl:element>
    </xsl:element>
  </xsl:template>

  <xsl:template match="keyHolder">
    <xsl:element name="Subjects">
```

```
      <xsl:element name="Subject">
        <xsl:element name="SubjectMatch">
          <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1.0
              :function:string-equal</xsl:attribute>
          <xsl:element name="AttributeValue">
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
            <xsl:value-of select="info/KeyName" />
          </xsl:element>
          <xsl:element name="SubjectAttributeDesignator">
            <xsl:attribute name="AttributeId">
                urn:oasis:names:tc:xacml:1.0:subject:subject-id</
                xsl:attribute>
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
          </xsl:element>
        </xsl:element>
      </xsl:element>
    </xsl:element>
  </xsl:template>

  <xsl:template match="play">
    <xsl:element name="Actions">
      <xsl:element name="Action">
        <xsl:element name="ActionMatch">
          <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1.0
              :function:string-equal</xsl:attribute>
          <xsl:element name="AttributeValue">
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
            play
          </xsl:element>
          <xsl:element name="ActionAttributeDesignator">
            <xsl:attribute name="AttributeId">
                urn:oasis:names:tc:xacml:1.0:action:action-id</
                xsl:attribute>
```

```
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
          </xsl:element>
        </xsl:element>
      </xsl:element>
    </xsl:element>
  </xsl:template>

  <xsl:template match="print">
    <xsl:element name="Actions">
      <xsl:element name="Action">
        <xsl:element name="ActionMatch">
          <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1.0
              :function:string-equal</xsl:attribute>
          <xsl:element name="AttributeValue">
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
            print
          </xsl:element>
          <xsl:element name="ActionAttributeDesignator">
            <xsl:attribute name="AttributeId">
                urn:oasis:names:tc:xacml:1.0:action:action-id</
                xsl:attribute>
            <xsl:attribute name="DataType">http://www.w3.org/2001/
                XMLSchema#string</xsl:attribute>
          </xsl:element>
        </xsl:element>
      </xsl:element>
    </xsl:element>
  </xsl:template>

  <xsl:template match="execute">
    <xsl:element name="Actions">
      <xsl:element name="Action">
        <xsl:element name="ActionMatch">
```

```
171          <xsl:attribute name="MatchId">urn:oasis:names:tc:xacml:1.0
                 :function:string-equal</xsl:attribute>
             <xsl:element name="AttributeValue">
               <xsl:attribute name="DataType">http://www.w3.org/2001/
                   XMLSchema#string</xsl:attribute>
               execute
             </xsl:element>
176          <xsl:element name="ActionAttributeDesignator">
               <xsl:attribute name="AttributeId">
                   urn:oasis:names:tc:xacml:1.0:action:action-id</
                   xsl:attribute>
               <xsl:attribute name="DataType">http://www.w3.org/2001/
                   XMLSchema#string</xsl:attribute>
             </xsl:element>
           </xsl:element>
181      </xsl:element>
       </xsl:element>
     </xsl:template>

     <xsl:template match="validityInterval">
186    <xsl:element name="Condition">
         <xsl:attribute name="FunctionId">urn:oasis:names:tc:xacml:1.0
             :function:and</xsl:attribute>
         <xsl:element name="Apply">
           <xsl:attribute name="FunctionId">urn:oasis:names:tc:xacml:1.0
               :function:dateTime-greater-than-or-equal</xsl:attribute>
           <xsl:element name="Apply">
191          <xsl:attribute name="FunctionId">urn:oasis:names:tc:xacml:1
                 .0:function:dateTime-one-and-only</xsl:attribute>
             <xsl:element name="EnvironmentAttributeSelector">
               <xsl:attribute name="DataType">http://www.w3.org/2001/
                   XMLSchema#dateTime</xsl:attribute>
               <xsl:attribute name="AttributeId">
                   urn:oasis:names:tc:xacml:1.0:environment:current-
                   dateTime</xsl:attribute>
             </xsl:element>
```

```
196         </xsl:element>
            <xsl:element name="AttributeValue">
              <xsl:attribute name="DataType">http://www.w3.org/2001/
                  XMLSchema#dateTime</xsl:attribute>
              <xsl:value-of select="notBefore" />
            </xsl:element>
201       </xsl:element>
          <xsl:element name="Apply">
            <xsl:attribute name="FunctionId">urn:oasis:names:tc:xacml:1.0
                :function:dateTime-less-than-or-equal</xsl:attribute>
            <xsl:element name="Apply">
              <xsl:attribute name="FunctionId">urn:oasis:names:tc:xacml:1
                  .0:function:dateTime-one-and-only</xsl:attribute>
206           <xsl:element name="EnvironmentAttributeSelector">
                <xsl:attribute name="DataType">http://www.w3.org/2001/
                    XMLSchema#dateTime</xsl:attribute>
                <xsl:attribute name="AttributeId">
                    urn:oasis:names:tc:xacml:1.0:environment:current-
                    dateTime</xsl:attribute>
              </xsl:element>
            </xsl:element>
211         <xsl:element name="AttributeValue">
              <xsl:attribute name="DataType">http://www.w3.org/2001/
                  XMLSchema#dateTime</xsl:attribute>
              <xsl:value-of select="notAfter" />
            </xsl:element>
          </xsl:element>
216     </xsl:element>
    </xsl:template>

    <xsl:template match="validityIntervalFloating">
      <xsl:element name="Condition">
221     <xsl:attribute name="FunctionId">urn:oasis:names:tc:xacml:1.0
            :function:integer-greater-than</xsl:attribute>
        <xsl:element name="AttributeValue">
```

```
        <xsl:attribute name="DataType">http://www.w3.org/2001/
            XMLSchema#integer</xsl:attribute>
        <xsl:value-of select="duration" />
      </xsl:element>
      <xsl:element name="AttributeValue">
        <xsl:attribute name="DataType">http://www.w3.org/2001/
            XMLSchema#integer</xsl:attribute>
        0
      </xsl:element>
    </xsl:element>
  </xsl:template>

  <xsl:template match="exerciseLimit">
    <xsl:element name="Condition">
      <xsl:attribute name="FunctionId">urn:oasis:names:tc:xacml:1.0
          :function:integer-greater-than</xsl:attribute>
      <xsl:element name="AttributeValue">
        <xsl:attribute name="DataType">http://www.w3.org/2001/
            XMLSchema#integer</xsl:attribute>
        <xsl:value-of select="count" />
      </xsl:element>
      <xsl:element name="AttributeValue">
        <xsl:attribute name="DataType">http://www.w3.org/2001/
            XMLSchema#integer</xsl:attribute>
        0
      </xsl:element>
    </xsl:element>
  </xsl:template>

  <xsl:template match="allConditions">
    <xsl:apply-templates select="validityInterval" />
    <xsl:apply-templates select="validityIntervalFloating" />
    <xsl:apply-templates select="exerciseLimit" />
  </xsl:template>
</xsl:stylesheet>
```

ANNEXE H

POLITIQUES D'EXEMPLE DU PROFILE MOBILE POUR ISO REL

Listing H.1 – iso-rel-mobile-profile-1.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<license>
  <grant>
    <keyHolder>
      <info>
        <KeyName>P800-g8NRYMG307NqJgmZ</KeyName>
      </info>
    </keyHolder>
    <execute/>
    <diReference>
      <identifier>urn:mobile:newsReader.app</identifier>
    </diReference>
    <exerciseLimit>
      <count>1</count>
    </exerciseLimit>
  </grant>
  <issuer>
    <keyHolder>
      <info>
        <KeyName>Content-provider-signing-key</KeyName>
      </info>
    </keyHolder>
  </issuer>
</license>
```

Listing H.2 – iso-rel-mobile-profile-2.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<license>
  <grant>
    <keyHolder>
```

```
      <info>
        <KeyName>PDA-Model-300-sc300NRG307NqJgmZ</KeyName>
      </info>
    </keyHolder>
    <play/>
    <diReference>
      <identifier>
          urn:mobile:travel:guides:ny:centralpark:34122948578946</
          identifier>
    </diReference>
    <validityInterval>
      <notBefore>2003-02-13T15:30:00</notBefore>
      <notAfter>2003-03-13T15:30:00</notAfter>
    </validityInterval>
  </grant>
  <grant>
    <keyHolder>
      <info>
        <KeyName>PDA-Model-300-sc300NRG307NqJgmZ</KeyName>
      </info>
    </keyHolder>
    <print/>
    <diReference>
      <identifier>
          urn:mobile:travel:guides:ny:centralpark:34122948578946</
          identifier>
    </diReference>
    <allConditions>
      <validityInterval>
        <notBefore>2003-02-13T15:30:00</notBefore>
        <notAfter>2003-03-13T15:30:00</notAfter>
      </validityInterval>
      <exerciseLimit>
        <count>1</count>
      </exerciseLimit>
    </allConditions>
```

```
  </grant>
  <issuer>
    <keyHolder>
      <info>
        <KeyName>Content-provider-signing-key</KeyName>
      </info>
    </keyHolder>
  </issuer>
</license>
```

Listing H.3 – request-P800-execute.xml

```
<license licenseId="request-P800-execute">
  <grant>
    <keyHolder>
      <info>
        <KeyName>P800-g8NRYMG307NqJgmZ/KeyName>
      </info>
    </keyHolder>
    <execute/>
    <diReference>
      <identifier>urn:mobile:newsReader.app</identifier>
    </diReference>
  </grant>
</license>
```

Listing H.4 – request-PDA-Model-300-play.xml

```
<license licenseId="request-PDA-Model-300-play">
  <grant>
    <keyHolder>
      <info>
        <KeyName>PDA-Model-300-sc300NRG307NqJgmZ</KeyName>
      </info>
    </keyHolder>
    <play/>
    <diReference>
```

```
      <identifier>
          urn:mobile:travel:guides:ny:centralpark:34122948578946
      </identifier>
    </diReference>
  </grant>
</license>
```

ANNEXE I

PROFILE CREATIVE COMMONS POUR ODRL 2.0

ODRL 2.0 Creative Commons Profile draft

Stéphane Tournié
stephane.tournie@info.fundp.ac.be

Jean-Noël Colin
jean-noel.colin@fundp.ac.be

May 27, 2012

Historical

version	changes
1.0	Initial version
1.1	CommercialUse as a purpose Constraint & few corrections
1.2	CommercialUse expressed by *commercialize* Action name, HighIncomeNationUse ccREL URL use, no more human-readable license link & additional information expressed with an Asset referenced by a Duty
1.3	human-readable license link described with a Dublin Core Description & Australian gov budget example added

1 Overview

In this document is defined a profile for the expression of Creative Commons licenses using the ODRL 2.0 language [1]. It aims to express the core Creative Commons semantics in ODRL 2.0, using already defined ODRL 2.0 Common Vocabulary and possibilities offer by the core model.

A profile has been defined earlier for the version 1.1 of ODRL [4] but since some elements of the language have been changed in version 2.0, defining a new Creative Commons profile for the new version of ODRL offers to meet the new semantics and vocabulary and to take advantage of these.

In the rest of this document, the acronym CC will be used instead of the term Creative Commons.

2 CC licenses characteristics

From the description of the CC licenses, we can highlight three main characteristics :

- Permissions - rights granted by the license
- Prohibitions - things prohibited by the license
- Requirements - restrictions imposed by the license

Each of these three characteristics contains specific values representing different Permissions, Prohibitions, and Requirements. The main values are summarized in table 1.

Table 1: main CC characteristics values

type	**value**	**description**
Permission	Reproduction	the work may be reproduced
Permission	Distribution	the work (and, if authorized, derivative works) may be distributed, publicly displayed, and publicly performed
Permission	DerivativeWorks	derivative works may be created and reproduced
Permission	Sharing	non commercial reproduction and distribution (like file-sharing) of the entire work are allowed
Prohibition	CommercialUse	rights may be exercised for commercial purposes
Prohibition	HighIncomeNationUse	rights may be exerciced in nations defined as high- income enconomies by the World Bank
Requirement	Notice	copyright and license notices must be kept intact
Requirement	Attribution	credit must be given to copyright holder and/or author
Requirement	ShareAlike	derivative works must be licensed under the same terms as the original work

3 CC Profile for ODRL 2.0

3.1 Mapping CC licenses characteristics values to ODRL 2.0 Common Vocabulary terms

ODRL 2.0 Common Vocabulary [2] already covers the semantic of the CC Requirement **Attribution**, with the *attribute* action entity name. To meet correctly the CC definition, this action entity name may be used in a Duty.

The semantic of the CC Prohibition **CommercialUse** is covered by the *commercialize* action entity name.

Then, almost all other values representing the CC Permissions, CC Prohibitions, and CC Requirements have been integrated as ODRL 2.0 Common Vocabulary terms too. So, for these values, there is no need to define new semantics. A direct mapping to the corresponding terms in the ODRL 2.0 Common Vocabulary may be done.

As action entity names defined in the ODRL 2.0 Common Vocabulary may be used with Permissions, Prohibitions, and Duties entities, we remind here an indication of how the characteristics values are defined to be used in CC description.

Table 2 resumes all CC values that can be mapped to ODRL 2.0 Common Vocabulary terms and in which case these terms may be used to meet the CC definition.

Table 2: mapping CC elements to ODRL 2.0 vocabulary terms

CC licenses value	**characteristic type**	**ODRL 2.0 vocabulary term**
Reproduction	Permission	copy reproduce
Distribution	Permission	distribute
DerivativeWorks	Permission	derive modify
Sharing	Permission	share
CommercialUse	Prohibition	commercialize
Attribution	Duty	attribute
Notice	Duty	attachPolicy
ShareAlike	Duty	shareAlike
SourceCode	Duty	attachSource

Since the ODRL 2.0 specification indicates that *"If several Permission entities refer to one Duty, then the Duty only has to be fulfilled once for all the Permission entities to become valid"* [1], a CC Requirement expressed as a Duty in ODRL 2.0 may be defined once in a Permission and then referenced by other Permissions and would have to be meet once for all. This rule corresponds naturally to the needs of the CC definition.

3.2 Addition to the ODRL 2.0 Common Vocabulary

The CC Prohibition **HighIncomeNationUse**, a constraint specific to the Developing Nations License [6], that prohibits to exercise rights of this license *"in nations defined as high-income enconomies by the World Bank "* cannot be mapped to an existing ODRL 2.0 Common Vocabulary term, as there is no term defined in the ODRL 2.0 Common Vocabulary with a semantic corresponding to this CC Prohibition. The solution retained here is to use the Prohibition term as defined for the Creative Commons Rights Expression Language [7], that matches exactly what we aim to express.

In ODRL 2.0, this new action entity name may be logically used in a Prohibition to meet correctly the CC definition.

3.3 Referencing human-readable CC Licenses

It can be useful and consistent to add a link from a ODRL expression of a valid licence to the corresponding human-readable description of the CC License. This may allow access to the human-readable version of the CC semantics, as well as linking the two

machine expressions.

This can be achieved using a **Dublin Core Metadata** [8] *Description* element.

Linking to the original CC License would be used only if the expressed license in ODRL 2.0 matches exactly a original CC license.

3.4 Expressing additional information

In the second part of the first example below, when the right holder wants to "*make clear the exact text that should be used for the Attribution requirement*", we define an Asset containing the text to be used and we make a reference to that (as a target) in the corresponding defined Duty.

In the second example below, the text used for the Attribution requirement is asked to be different regarding to the used made of the ressource (modified or not). We use in this case some Dublin Core Metadata entities to define the asset referenced in the Duty.

Obviously, there may be a lot of different cases where we may need to add additional information about an entity. Dublin Core or other metadata standards entities can be used to express those semantics within ODRL 2.0 licences expressions.

4 Example Scenario

4.1 Original CC project website example

We are using in this section an example originally given on the CC project website. There is no more active hypertext link to this example but in the aim to express a sample CC license using the CC profile for ODRL 2.0, the basic statements included here are sufficient to understand the meaning.

In the following license samples, solely additional information about the right holder Party are not expressed.

"Anita chooses the Attribution-NonCommercial-ShareAlike CC license."

```
<?xml version="1.0" encoding="UTF-8"?>
<policy xmlns="http://w3.org/ns/odrl/2/"
  type="http://w3.org/ns/odrl/vocab#offer"
  uid="urn:anita:volcanolove:01">
  <dcds:description xmlns:dcds="http://purl.org/dc/xmlns/2008/09/01/dc-ds-
      xml/">
    <dcds:statement dcds:propertyURI="http://purl.org/dc/terms/source"
                    dcds:valueURI="http://creativecommons.org/licenses/by-nc-sa
                        /2.0">
```

```
    <dcds:valueString>Attribution-NonCommercial-ShareAlike 2.0 Generic (CC
        BY-NC-SA 2.0)</dcds:valueString>
    </dcds:statement>
  </dcds:description>
  <permission>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3"/>
    <action name="http://w3.org/ns/odrl/vocab#copyReproduce"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="d1">
          <action name="http://w3.org/ns/odrl/vocab#attachPolicy"/>
        </duty>
    <duty uid="d2">
          <action name="http://w3.org/ns/odrl/vocab#shareAlike"/>
        </duty>
    <duty uid="d3">
          <action name="http://w3.org/ns/odrl/vocab#attribute"/>
        </duty>
  </permission>
  <permission>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3"/>
    <action name="http://w3.org/ns/odrl/vocab#distribute"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </permission>
  <permission>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3"/>
    <action name="http://w3.org/ns/odrl/vocab#deriveModify"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </permission>
  <prohibition>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3"/>
    <action name="http://w3.org/ns/odrl/vocab#commercialize"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </prohibition>
</policy>
```

Listing 1: cc-odrl20-sample1.xml

"Extending the above example, lets say that Anita wants to make two additions. Firstly she wants to limit the Distribution permission to the USA only. Secondly, she wants to

make clear the exact text that should be used for the Attribution requirement."

```xml
<?xml version="1.0" encoding="UTF-8"?>
<policy xmlns="http://w3.org/ns/odrl/2/"
  type="http://w3.org/ns/odrl/vocab#offer"
  uid="urn:anita:volcanolove:02">
  <permission>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3"/>
    <action name="http://w3.org/ns/odrl/vocab#copyReproduce"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="d1">
          <action name="http://w3.org/ns/odrl/vocab#attachPolicy"/>
        </duty>
    <duty uid="d2">
          <action name="http://w3.org/ns/odrl/vocab#shareAlike"/>
        </duty>
    <duty uid="d3">
          <action name="http://w3.org/ns/odrl/vocab#attribute"/>
      <asset uid="http://example.com/text:01" relation="http://w3.org/ns/
          odrl/vocab#target"/>
        </duty>
  </permission>
  <permission>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3"/>
    <action name="http://w3.org/ns/odrl/vocab#distribute"/>
    <constraint name="http://w3.org/ns/odrl/vocab#spatial" operator="http:
        //w3.org/ns/odrl/vocab#eq" rightOperand="urn:iso3166:US"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </permission>
  <permission>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3"/>
    <action name="http://w3.org/ns/odrl/vocab#deriveModify"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </permission>
  <prohibition>
    <asset uid="urn:anita-music:songs:volcanoLove.mp3"/>
    <action name="http://w3.org/ns/odrl/vocab#commercialize"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </prohibition>
</policy>
```

Listing 2: cc-odrl20-sample2.xml

In Listing 2, we follow the guidelines from the profile for ODRL 1.1 : *"Since the above Distribution permission is narrower than the original CC license, Anita removes the link from the reference element to the original CC license."*.

"Next, Anita wishes to make the lyrics to her song available. Anita chooses the Attribution-NonCommercial-NoDerivs CC license as this is the closest. However, she decides that she only wants the public to print her lyrics no more than once, so she includes the Display and Print permission with a Count constraint. She does not include the Reproduction permission and does not include a Reference to the CC License (as the semantics are not the same)."

```
<?xml version="1.0" encoding="UTF-8"?>
<policy xmlns="http://w3.org/ns/odrl/2/"
  type="http://w3.org/ns/odrl/vocab#offer"
  uid="urn:anita:lyrics:volcanolove:01">
  <permission>
    <asset uid="urn:anita-music:lyrics:volcanoLove.pdf"/>
    <action name="http://w3.org/ns/odrl/vocab#distribute"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="d1">
          <action name="http://w3.org/ns/odrl/vocab#attachPolicy"/>
        </duty>
    <duty uid="d2">
          <action name="http://w3.org/ns/odrl/vocab#shareAlike"/>
        </duty>
    <duty uid="d3">
          <action name="http://w3.org/ns/odrl/vocab#attribute"/>
        </duty>
  </permission>
  <permission>
    <asset uid="urn:anita-music:lyrics:volcanoLove.pdf"/>
    <action name="http://w3.org/ns/odrl/vocab#display"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </permission>
  <permission>
    <asset uid="urn:anita-music:lyrics:volcanoLove.pdf"/>
    <action name="http://w3.org/ns/odrl/vocab#print"/>
    <constraint name="count" operator="eq" rightOperand="1"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
```

```
  </permission>
  <prohibition>
    <asset uid="urn:anita-music:lyrics:volcanoLove.pdf"/>
    <action name="http://w3.org/ns/odrl/vocab#commercialize"/>
    <party uid="urn:music-people:id-anita-001" function="http://w3.org/ns/
        odrl/vocab#assigner"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
    <duty uid="#d3"/>
  </prohibition>
</policy>
```

Listing 3: cc-odrl20-sample3.xml

4.2 Australian gov budget example

The second example given here remain from the Australian government. It aims to describe in ODRL the licence associated with a budget document [9] available for use under a Creative Commons BY Attribution 3.0 Australia licence [10].

```
<?xml version="1.0" encoding="UTF-8"?>
<policy xmlns="http://w3.org/ns/odrl/2/"
  type="http://w3.org/ns/odrl/vocab#offer"
  uid="urn:budget:gov:au:01">
  <permission>
    <asset uid="http://budget.gov.au/2012-13"/>
    <action name="http://w3.org/ns/odrl/vocab#share"/>
    <party uid="http://www.budget.gov.au" function="http://w3.org/ns/odrl/
        vocab#assigner"/>
    <duty uid="d1">
          <action name="http://w3.org/ns/odrl/vocab#attachPolicy"/>
        </duty>
    <duty uid="d2">
          <action name="http://w3.org/ns/odrl/vocab#attribute"/>
          <asset uid="http://www.budget.gov.au/attribute#asSupplied"
              relation="http://w3.org/ns/odrl/vocab#target"/>
        </duty>
  </permission>
  <permission>
    <asset uid="http://budget.gov.au/2012-13"/>
    <action name="http://w3.org/ns/odrl/vocab#deriveModify"/>
    <party uid="http://www.budget.gov.au" function="http://w3.org/ns/odrl/
        vocab#assigner"/>
    <duty uid="#d1"/>
    <duty uid="d3">
          <action name="http://w3.org/ns/odrl/vocab#attribute"/>
          <asset uid="http://www.budget.gov.au/attribute#modified" relation
              ="http://w3.org/ns/odrl/vocab#target"/>
        </duty>
  </permission>
  <permission>
    <asset uid="http://budget.gov.au/2012-13"/>
```

```
    <action name="http://w3.org/ns/odrl/vocab#commercialize"/>
    <party uid="http://www.budget.gov.au" function="http://w3.org/ns/odrl/
        vocab#assigner"/>
    <duty uid="#d1"/>
    <duty uid="#d2"/>
  </permission>
</policy>

<!-- External Asset -->
<dcx:description xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:dcx="http://purl.org/dc/xml/"
          dcx:resourceURI="http://www.budget.gov.au/attribute#asSupplied">
    <dc:title>
    <dcx:valueString>Source: The Treasury</dcx:valueString>
  </dc:title>
</dcx:description>

<!-- External Asset -->
<dcx:description xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:dcx="http://purl.org/dc/xml/"
         dcx:resourceURI="http://www.budget.gov.au/attribute#modified">
    <dc:title>
    <dcx:valueString>Based on Treasury data</dcx:valueString>
  </dc:title>
</dcx:description>
```

Listing 4: budget-gov-au.xml

Bibliography

[1] ODRL Community Group, www.w3.org/community/odrl/two/model

[2] ODRL Community Group, www.w3.org/community/odrl/two/vocab

[3] ODRL Community Group, www.w3.org/community/odrl/two/xml

[4] ODRL Community Group, odrl.net/Profiles/CC/SPEC.html

[5] http://creativecommons.org/licenses

[6] http://creativecommons.org/licenses/devnations/2.0

[7] http://creativecommons.org/ns#HighIncomeNationUse

[8] http://dublincore.org/documents/dcmes-xml

[9] http://budget.gov.au/2012-13/content/bp1/html/bp1_prelims.htm

[10] http://creativecommons.org/licenses/by/3.0/au/deed.en

1 Annexe

1.1 Profile dictionary

Table 3: Profile dictionary

Type	**Identifier**	**Definition**
- action entity name - URL to be used : http://creativecommons.org/ns#HighIncomeNationUse	HighIncomeNationUse	Act to exercise rights in nations defined as high-income economies by the World Bank