Découverte automatisée de hiérarchies de rôles pour les politiques de contrôle d'accès

Romuald Thion, Stéphane Coulondre

INSA-Lyon
Laboratoire LIRIS (UMR 5205)
Bât. Blaise Pascal (501)
20, Av. A. Einstein
69621 Villeurbanne CEDEX
romuald.thion@insa-lyon.fr

RÉSUMÉ. *L'identification des rôles dans un système – l'ingénierie des rôles – est une tâche essentielle dans l'établissement des politiques de contrôle d'accès à rôles (*RBAC*). La stratégie classique d'identification débute par une analyse fine du métier, à partir de laquelle sont dérivés les rôles. Cette approche, longue et coûteuse, impose une grande expertise du domaine et ne tire aucun profit des privilèges existants. Pour palier à ces défauts, une stratégie dite* role-mining *a été proposée. Basée sur l'agrégation des permissions existantes, cette approche automatise le processus d'ingénierie des rôles.*

*Cet article propose une technique d'ingénierie des rôles hiérarchisés implicitement présents dans les droits existants. Notre proposition est fondée sur l'identification de la notion de rôle à celle de concept formel. Nous pouvons ainsi caractériser formellement le processus d'ingénierie et mettre à profit les travaux sur l'analyse de concepts formel (*FCA*). La fertilisation croisée entre ces deux problèmes ouvre des perspectives sur la formalisation du contrôle d'accès.*

ABSTRACT. *Role-engineering is the task of discovering roles in a system. This task is essential in building efficient role-based access control policies. The classical approach in role-engineering is top-down. It relies on business process analysis to define roles from basic tasks, but ignore existing privileges. This approach is time consuming and expensive.*

This article presents a new automated role-engineering technique – a role-mining technique. The paradigm used is formal concept analysis, which aim at extracting implicit lattice from binary relation.

MOTS-CLÉS : *Ingénierie des rôles, contrôle d'accès à base de rôles, sous-hiérarchie de Galois*

KEYWORDS: *Role-engineering, Role-Based Access Control, Galois Sub-Hierarchy*

1re soumission à *ISI*, le 24 janvier 2008

1. Contrôle d'accès aux systèmes d'information

1.1. *Politique de sécurité*

De nombreux dispositifs et pratiques participent à la sécurité des systèmes d'informations : la protection des réseaux , le cryptage de l'information, la sauvegarde des données, la redondance matérielle et logicielle ou encore les architectures d'authentification. Parmi les moyens mis en œuvre pour renforcer la sécurité, les *politiques de sécurité* forment un dispositif organisationnel et technique fondamental pour garantir la sécurité de l'information.

Cette politique peut être spécialisée selon les propriétés auxquelles elle s'attache plus spécifiquement en politiques de sécurité *physique*, *administrative* ou *logique*. La sécurité logique concerne deux aspects principaux :

– l'identification et l'authentification : donner et prouver son identité,

– l'*autorisation*, ou *contrôle d'accès* : la vérification de la légitimité des opérations demandées selon une politique de contrôle d'accès.

Cet article s'intéresse aux politiques de sécurité logique et plus précisément à l'organisation des droits et au contrôle des accès aux systèmes. Les premières formalisations du contrôle d'accès sont dues à Lampson, qui parmi les premiers a défini les concepts de *sujet*, d'*objet* et d'*actions* (Lampson, 1974). Le modèle de Lampson est la *matrice de contrôle d'accès* qui permet de représenter le triplet d'autorisation entre sujets ($\mathcal{S}$), actions ($\mathcal{A}$) et objets ($\mathcal{O}$), figure 1.

Définition (Contrôle d'accès (Lampson, 1974))**.** *Le contrôle d'accès est un mécanisme grâce auquel un système autorise ou interdit les* actions *demandées par des* sujets *sur des* objets*.*

1.2. *Contrôle d'accès à rôles*

Les modèles de *contrôle d'accès à rôles* (*Role-Based Access Control* – RBAC) sont apparus à partir des années 90 pour palier les difficultés d'administration rencontrées avec les modèles matriciels. Le principe général de RBAC est d'introduire un niveau d'*indirection* entre utilisateurs et permissions. Le concept de *rôle* est défini comme (Sandhu *et al.,* 1996) :

> *. . . a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role.*

La principale motivation de l'introduction de la notion de rôle dans l'organisation des droits est la constatation qu'une grande partie des décisions de contrôle d'accès sont déterminées par l'*autorité hiérarchique* et la *fonction* des utilisateurs. Utiliser le rôle comme intermédiaire entre utilisateurs et permissions facilite et simplifie les tâches d'administration en diminuant le nombre d'affectations à manipuler.

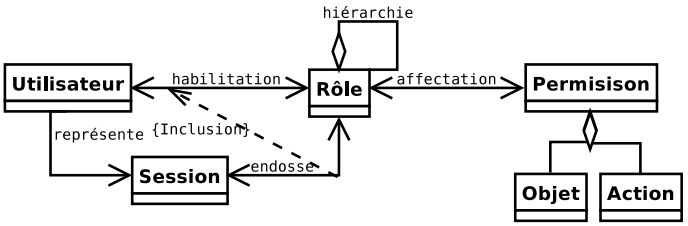


Figure 1. *Représentation* UML *du modèle* RBAC_1

La hiérarchisation des rôles permet de calquer la structure les droits sur celle des organisations. La figure 1 donne une représentation UML du modèle à rôles hiérarchisés RBAC_1, un des principaux modèles de la famille RBAC. Les modèles RBAC sont largement adoptés par les entreprises et les industriels pour tous types de systèmes (CERT/CC *et al.,* 2005). On les retrouve dans les logiciels Trusted Solaris, Windows Authorization Manager, Oracle 9 et Sybase Adaptive Server par exemple.

1.3. *Ingénierie des rôles*

Un des arguments principaux en faveur d'une gestion des droits basée sur des rôles hiérarchisés est que les coûts de l'administration des politiques sont réduits (Crampton, 2003, Ferraiolo *et al.,* 2003). La hiérarchisation permet en effet de réduire la taille des politiques et proposant des affectations implicites déduites à partir des propriétés algébriques de la relation d'héritage entre rôles.

Le travail d'Edward Coyne est à l'origine du terme *ingénierie des rôles* (Coyne, 1996). Sa proposition centre l'activité d'ingénierie comme la définition des affectations de permissions aux rôles, sous-entendant aussi celle de rôles aux utilisateurs :

> *The concept of role engineering (RE) is an approach to defining roles and assigning permissions to the roles. RE must capture the organization's business rules, as these relate to access control, ...*

L'ingénierie des rôles est entreprise lors de la conception d'une politique de contrôle d'accès RBAC. Si l'ingénierie des rôles est une activité essentielle pour que l'établissement d'une politique RBAC soit fructueux, c'est également l'activité la plus coûteuse et la plus longue. Un rapport du NIST indique que le coût d'identification de chaque rôle est évalué à environ 5.000 dollars (Gallaher *et al.,* 2002).

La thèse de Pete Epstein a considérablement étendu Coyne (Epstein, 2002). L'activité d'ingénierie des rôles définie Epstein peut être résumée par « l'identification d'un ensemble de rôles qui soit *complet*, *correct* et *efficace* ». Deux principales stratégies d'identification des rôles ont été proposées, illustrées par la figure 2 :

– l'approche *descendante* est basée sur la décomposition des rôles en concepts intermédiaires. Cette approche se base sur la structure de l'organisation, mais pas

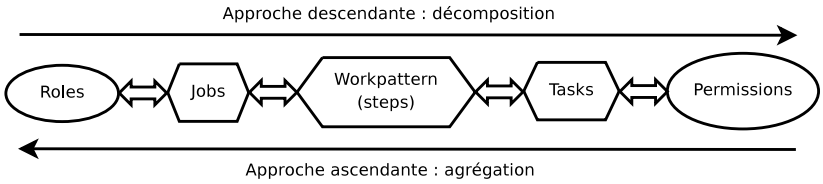


Figure 2. *Deux approches d'ingénierie des rôles (Epstein, 2002)*

sur les privilèges existants. Elle a été appliquée avec succès dans de grandes structures (Roeckle *et al.,* 2000), et étendue par des méthodes à base de scénarios et de cas d'utilisation (Neumann *et al.,* 2002). Cette approche, basée sur une modélisation rigoureuse des processus métiers, n'a pas pas à notre connaissance été automatisée. Elle est adaptée aux organisations qui déploient un modèle RBAC *ex nihilo*,

– l'approche *ascendante*, dite *role-mining* (Kuhlmann *et al.,* 2003) est une identification des rôles implicitement présents dans le système. Cette approche est basée sur l'agrégation des permissions existantes, sans tenir compte cependant de la structure de l'organisation. Cette approche se base sur des droits existants.

1.4. *Objectifs*

Il est rare qu'une organisation mette en place un modèle de contrôle d'accès RBAC sans se baser sur des droits existants. Un des objectifs des travaux en ingénierie des rôles est d'automatiser l'approche ascendante d'identification des rôles et de leurs hiérarchies. Le but est de faciliter et de réduire la durée et le coût de l'ingénierie des rôles. La proposition de cet article va dans ce sens en proposant une méthode ascendante d'identification automatisée de rôles.

Nous considérerons les modèles de contrôle d'accès comme des *moyens* pour structurer des politiques. L'*objectif* de tout modèle de contrôle d'accès est de permettre la dérivation du triplet fondamental $\mathcal{ACCESS} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{O}$. Nous envisageons ansi l'activité d'ingénierie des rôles comme une activité de rétroconception qui tâche d'identifier des relations intermédiaires dans le triplet $\mathcal{ACCESS}$.

La notion de rôle dans les modèles RBAC diffère de la notion de groupe : un groupe est essentiellement une collection d'utilisateurs alors que le rôle est *à la fois* une collection d'utilisateurs *et* de permissions (Sandhu *et al.,* 1996). C'est donc un concept qui peut être défini soit par son intention – les permissions dont dispose le rôle – soit par son extension – les utilisateurs qui l'endossent. Cette définition nous sert de base pour notre démarche. Nous allons assimiler la notion de *rôle* dans les modèles RBAC à celle de *concept formel* de l'analyse de concepts formels (*Formal Concept Analysis* – FCA).

	1	2	3	4
Alice	rw	r	r	
Bob	r	rw	r	rwx
Charly	r	r	rw	rwx
Denise			r	r

Tableau 1. *Matrice de contrôle d'accès*

Ainsi, nous mettons en correspondance la formalisation des modèles de contrôle d'accès RBAC et celle de la FCA. Basés sur la correspondance entre rôle concept formel, nous développons le parallèle entre ces deux domaines jusqu'à parvenir à définir l'ingénierie des rôles en terme d'extraction de concepts formels.

1.5. *Structure de l'article*

La section suivante va faire correspondre la matrice de contrôle d'accès de Lampson à la notion de contexte formel (section 2). Ensuite, nous introduirons les rôles dans le contrôle d'accès puis leur hiérarchisation. Ces deux notions sont mises en correspondance avec celle concept formel et de treillis des concepts (section 3).

Grâce à ces deux étapes préliminaires, nous proposerons, en section 4, une définition de l'ingénierie des rôles exprimée comme l'extraction d'un ensemble partiellement ordonné de concepts formels dans une relation binaire existante. Les liens dont nous avons besoin entre RBAC et FCAétant tissés, nous décrivons la méthode algorithmique sélectionnée en section 5.

Enfin, les deux dernières sections comparent les résultats obtenus par notre méthode aux travaux existants puis concluent notre proposition. Nous essaierons d'interpréter les résultats obtenus par l'extraction de concepts en termes de structuration des droits et de les mettre en perspective avec d'autres définitions de l'ingénierie des rôles. Les définitions concernant la FCA sont issues de (Wille, 1997) en utilisant la notation de (Arévalo *et al.,* 2007). Celles qui concernent le contrôle d'accès sont issues de (Ferraiolo *et al.,* 2003, Sandhu *et al.,* 1996) en prenant en compte les critiques qui leur ont été adressées (Li *et al.,* 2007).

2. Matrice de contrôle d'accès et contexte formel

2.1. *Matrice de contrôle d'accès*

La matrice de contrôle d'accès introduite par Lampson propose une formalisation algébrique simple du contrôle d'accès basée sur un triplet fondamental $\mathcal{ACCESS} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{O}$) faisant intervenir :

– l'ensemble $\mathcal{S}$ des *sujets* connus du système. Les sujets peuvent être des processus, des machines : toute entité informatique qui représente l'utilisateur au sein du système et se comporte selon sa volonté.

– l'ensemble $\mathcal{O}$ des *objets* connus du système. Un objet est une entité du système comme un tuple, une table (cas des systèmes de gestion de bases de données), un objet ou un fichier (pour le cas des systèmes de fichier),

– l'ensemble $\mathcal{A}$ des *actions* connues du système que l'on peut effectuer sur les objets. Les actions comprennent par exemple les opérations de sélection, suppression, modification et création de tuples dans un système de gestion de bases de données,

Une *session* est l'entité qui représente un utilisateur *actif* au sein du système. Dans une session donnée, un utilisateur choisit d'endosser tout ou partie des rôles qui lui sont attribués. Pour notre application, nous confondons la notion de *sujet* de Lampson ($\mathcal{S}$) à celle d'*utilisateur* dans les modèles RBAC ($\mathcal{U}$). Nous nous permettons cette simplification, car nous nous intéressons seulement à la hiérarchie de rôle qui est un des aspects *statiques* des politiques RBAC, c'est-à-dire ceux sur lesquels les utilisateurs ne disposent d'aucun droit et qui ne dépendent pas de l'exécution du système.

Le tableau 1 est un exemple jouet de matrice de contrôle d'accès, impliquant quatre sujets (en lignes), quatre objets (en colonnes) et trois actions (`r` pour *read*, `w` pour *write* et `x` pour *execute*). De nombreux systèmes (systèmes d'exploitation ou de gestion de base de données) ont mis en œuvre le contrôle d'accès basé sur des matrices.

2.2. *Restriction à une relation binaire*

Tout d'abord, nous décomposons la relation ternaire $\mathcal{ACCESS}$ en deux relations à l'aide de la notion de permission. L'ensemble des permissions $\mathcal{P}$ est défini en intention par les paires d'actions et d'objets utilisées dans la politique de contrôle d'accès. On définit ainsi une relation $\mathcal{UP} \in \mathcal{U} \times \mathcal{P}$ qui est une restriction de la relation $\mathcal{ACCESS}$. En effet, dans les modèles RBAC, on n'affecte jamais directement d'action ou d'objet à un rôle, on affecte des couples d'action et d'objet, c'est à dire des permissions.

$$(u, a, o) \in \mathcal{ACCESS} \Leftrightarrow \exists p \in \mathcal{P}, p = (a, o) \wedge (u, p) \in \mathcal{UP}$$

L'intérêt de cette transformation est de pouvoir ensuite manipuler une matrice booléenne, car de nombreux problèmes de fouille de données ont été caractérisés sur de telles structures et de nombreux algorithmes sont disponibles. De plus, dans plusieurs cas d'utilisation on a seulement accès aux identifiants de permissions et pas à leur décomposition en action et objet.

Le tableau 2 présente graphiquement la relation $\mathcal{UP}$ construite à partir de la relation $\mathcal{ACCESS}$ du tableau 1. Nous avons pris comme identifiant de chaque permission la concaténation des identifiants d'action et d'objet. Ainsi, `r1` indique le droit de lecture sur le fichier `1`. Cette relation $\mathcal{UP}$ est interprétée dans notre approche, soit comme une restriction de la matrice de Lampson, soit comme un contexte formel.

	r1	w1	r2	w2	r3	w3	r4	w4	x4
Alice	×	×	×		×				
Bob	×		×	×	×		×	×	×
Charly	×		×		×	×	×	×	×
Denise					×		×		

Tableau 2. *Contexte formel construit à partir de $\mathcal{ACCESS}$*

2.3. *Contexte formel*

La formalisation de l'analyse de concepts formels part d'un *contexte formel* : une relation *binaire* entre objets et attributs. L'opération d'analyse consiste, à partir d'un contexte formel, à identifier une hiérarchie de concepts aux propriétés particulières (Wille, 1997).

Définition (Contexte formel et application α et ω)**.** *Un* contexte formel *est un triplet $\mathbb{K} = (O, A, J)$ où O est l'ensemble des objets, A l'ensemble des attributs et $J \subseteq O \times A$ la relation d'incidence. La relation d'incidence J induit deux applications α et ω qui relient*[1] 2^O *et* 2^A.

$$\forall X \in 2^O, \alpha(X) = \{y \in A \mid \forall\, x \in X, (x, y) \in J\}$$
$$\forall Y \in 2^A, \omega(Y) = \{x \in O \mid \forall\, y \in Y, (x, y) \in J\}$$

Les applications α et ω définissent une correspondance de Galois *entre 2^O et 2^A. Les applications $\alpha \circ \omega$ et $\omega \circ \alpha$ définissent deux opérateurs de fermetures sur 2^O et 2^A respectivement.*

Pour l'application à découverte de rôles, les *objets* de la FCA sont des *utilisateurs* et les *attributs* des permissions. La relation d'incidence indique quels sont les droits dont les utilisateurs disposent. Nous pouvons ainsi décrire le contexte formel $\mathbb{K}$: ce serait la matrice des droits d'accès du système, la relation $\mathcal{UP}$ du tableau 2.

Les applications α (resp. ω) s'interprètent alors comme l'application qui à un ensemble d'utilisateurs (resp. d'permissions) fait correspondre l'ensemble des permissions (resp. des utilisateurs) avec lesquels *tous* les utilisateurs (resp. permissions) de l'ensemble sont en relation.

3. Rôles hiérarchisés et treillis des concepts formels

La famille des sous-ensembles fermés de O et de A, notés C^o et C^a, présentent deux propriétés fondamentales de la FCA : α et ω sont des bijections entre ces familles,

1. On note par 2^X l'ensemble des sous-ensembles de X.

	Notation	Description
Concepts	$\mathcal{U}$	ensemble fini d'utilisateurs
	$\mathcal{R}$	ensemble fini de rôles
	$\mathcal{A}$	ensemble fini d'actions
	$\mathcal{O}$	ensemble fini d'objets
Relations	$\mathcal{URA} \subseteq \mathcal{U} \times \mathcal{R}$	affectation de rôles aux utilisateurs
	$\mathcal{PRA} \subseteq \mathcal{R} \times \mathcal{A} \times \mathcal{O}$	affectation de permissions aux rôles
	$\mathcal{RH} \subseteq \mathcal{R} \times \mathcal{R}$	relation d'héritage $\succeq$ entre rôles
	$\mathcal{ACCESS} \subseteq \mathcal{U} \times \mathcal{A} \times \mathcal{O}$	le triplet d'autorisation

Tableau 3. *Formalisation ensembliste de* RBAC *(Ferraiolo* et al., *2003)*

appelées *correspondances de Galois*. De plus, ces familles munies de l'inclusion ensembliste forment deux treillis complets isomorphes, appelés *treillis de Galois*. Pour notre application cela signifie que grâce à α et ω, on va pouvoir identifier des couples d'utilisateurs et de permissions (u, p) qui sont organisés selon un treillis. Avec (u, p) et la définition que nous avons donnée de $\mathcal{P}$ nous pouvons retrouver triplet (u, a, o).

3.1. *Hiérarchie de rôles dans les modèles* RBAC

La hiérarchie des rôles est une des innovations proposées dans les modèles RBAC. Elle permet de réduire le nombre de rôles et d'affectations (entre rôles et utilisateurs, ou entre rôles et permissions) en introduisant une relation binaire d'héritage entre rôles. Selon les propriétés algébriques cette relation respecte, différents types de hiérarchies ont été identifiés : limitée en arbres ou générale (Ferraiolo *et al.,* 2003). Nous nous intéressons au cas général où la hiérarchie est un ordre partiel sur les rôles. La formalisation ensembliste de RBAC est résumée par le tableau 3.

Définition (Hiérarchie générale de rôles)**.** *$\mathcal{RH} \subseteq \mathcal{R} \times \mathcal{R}$ est une relation transitive, réflexive et antisymétrique appelée* relation d'héritage*, noté $\succeq$.*

Soient deux rôles $r_1 \in \mathcal{R}$ et $r_2 \in \mathcal{R}$ tels que $r_1 \succeq r_2$. r_1 est dit plus spécifique *que r_2 et r_2 est dit* plus général *que r_1. On dit aussi que r_1 hérite de r_2.*

L'introduction de hiérarchies de rôles conduit à des affectations implicites. En effet, quand $a \succeq b$ et qu'un utilisateur u est *explicitement* affecté au rôle a ($(u, r) \in \mathcal{URA}$), cet utilisateur est *implicitement* affecté au rôle b. Cette nuance entre affectations explicite et implicite est représentée par les applications $assign\_users$ et $assign\_perms$ (affectés) ainsi que par $auth\_users$ et $auth\_perms$ (autorisés).

Définition (Utilisateurs affectés et autorisés)**.** *L'application $assign\_users : \mathcal{R} \rightarrow 2^{\mathcal{U}}$ fait correspondre à un rôle l'ensemble des utilisateurs qui lui sont explicitement*

affectés. *L'application* $auth\_users : \mathcal{R} \rightarrow 2^{\mathcal{U}}$ *fait correspondre à un rôle l'ensemble des utilisateurs* autorisés *à l'endosser :*

$$assign\_users(r) = \{u \in \mathcal{U} \mid (u, r) \in \mathcal{URA}\}$$
$$auth\_users(r) = \{u \in \mathcal{U} \mid r' \succeq r \wedge (u, r') \in \mathcal{URA}\}$$

Définition (Permissions affectées et autorisées)**.** *L'application* $assign\_perms : \mathcal{R} \rightarrow 2^{\mathcal{A} \times \mathcal{O}}$ *fait correspondre à un rôle l'ensemble des permissions qui lui sont explicitement* affectées*. L'application* $auth\_perms : \mathcal{R} \rightarrow 2^{\mathcal{A} \times \mathcal{O}}$ *fait correspondre à un rôle l'ensemble des permissions* autorisées *:*

$$assign\_perms(r) = \{(a, o) \in \mathcal{A} \times \mathcal{O} \mid (r, a, o) \in \mathcal{PRA}\}$$
$$auth\_perms(r) = \{(a, o) \in \mathcal{A} \times \mathcal{O} \mid r' \preceq r \wedge (r', a, o) \in \mathcal{PRA}\}$$

3.2. *Treillis des concepts formels*

Définition (Concept formel et treillis)**.** *Une paire d'ensembles en correspondance* $c = (X, Y)$ *tels que* $Y = \alpha(X)$ *(ou* $X = \omega(Y)$*) est appelée* concept formel*.*

L'ensemble des concepts C *muni de la relation d'ordre partiel* $\leq_C$*:* $(X_1, Y_1) \leq_C (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2$ *(ou* $Y_2 \subseteq Y_1$*) forme un treillis de Galois appelé* treillis de concepts*.*

L'analyse de concepts formels définit les concepts comme des ensembles *maximaux* d'objets partageant les mêmes attributs. Chaque concept $c = (X, Y)$ définit ainsi deux relations : une première entre objets et concept et une seconde entre concept et attributs. L'ensemble X des objets d'un concept est appelé *extension* (complète), l'ensemble Y des attributs est appelé *intention* (complète).

Théorème (Théorème fondamental de la FCA)**.** *L'ensemble* C *de tous les concepts qui peuvent être construits sur un contexte formel* $\mathbb{K} = (O, A, J)$ *donné forme un* treillis complet *quand il est ordonné par l'inclusion ensembliste.*

Appliqué au contrôle d'accès, le treillis[2] des concepts définit deux relations entre utilisateurs et concepts d'une part, et concepts et permissions d'autre part. Ces deux relations correspondent respectivement aux relations $\mathcal{URA}$ et $\mathcal{PRA}$, qui relient rôles et utilisateurs et rôles et permissions. L'extension d'un rôle est l'ensemble de ses utilisateurs *autorisés*, son intention ses permissions *autorisées*. La relation d'ordre $\leq_C$ est considérée comme un ordre partiel entre les rôles : la relation d'héritage $\mathcal{RH}$.

3.3. *Mise en correspondance des applications*

Pour l'application que nous proposons, nous utilisons indifféremment *utilisateur* pour *objet* ou *permission* pour *attribut*, selon que l'on s'exprime en termes *d'ingénie-*

2. Un treillis est un ensemble partiellement ordonné tel que chaque couple d'éléments admette une borne supérieure et une borne inférieure.

Analyse de concepts	Contrôle d'accès à rôles
concept	rôle
objet	utilisateur
attribut	permission
relation d'ordre partiel	relation d'héritage
extension réduite	utilisateurs *explicitement* affectés
intention réduite	permissions *explicitement* affectées
extension complète	utilisateurs *implicitement* affectés
intention complète	permissions *implicitement* affectées
concepts ancêtres	rôles dont hérite un rôle
concepts descendants	rôles héritant d'un rôle

Tableau 4. *Identification des termes de* FCA *et de* RBAC

rie des rôles ou *d'analyse de concepts formels*. En prolongeant la correspondance, on peut exprimer les quatre applications définies avec la hiérarchie RBAC avec les termes de la FCA. Pour un rôle r donné :

– $assign\_users : \mathcal{R} \rightarrow 2^{\mathcal{U}}$, fait correspondre au rôle r son l'*extension réduite*,

– $auth\_users : \mathcal{R} \rightarrow 2^{\mathcal{U}}$, fait correspondre au rôle r son *extension complète*,

– $assign\_perms : \mathcal{R} \rightarrow 2^{\mathcal{P}}$, fait correspondre au rôle r son *intention réduite*,

– $auth\_perms : \mathcal{R} \rightarrow 2^{\mathcal{P}}$, fait correspondre au rôle r son *intention complète*.

Nous arrivons ainsi à la définition d'un *rôle formel* : un couple d'utilisateurs et d'attributs en correspondance par les applications $auth\_users$ et $auth\_perms$. Le tableau 4 résume les principales correspondances que nous définissons et mettons en œuvre entre RBAC et FCA.

4. Analyse de concepts formels et ingénierie des rôles

L'intérêt des concepts intermédiaires entre utilisateurs et permissions est de permettre le regroupement de privilèges. Le tableau 2 présente en grisé le regroupement des permissions `r1`, `r2` et `r3`, dont les utilisateurs `Alice`, `Bob` et `Charly` disposent tous les trois. Ces groupements d'utilisateurs $X = \{\texttt{Alice}, \texttt{Bob}, \texttt{Charly}\}$ et de permissions $Y = \{\texttt{r1}, \texttt{r2}, \texttt{r3}\}$ sont tels que $\alpha(X) = Y$ et $\alpha(Y) = X$. Le couple (X, Y) est par définition un *concept formel*.

Dans un modèle de contrôle d'accès RBAC, ce regroupement est un *rôle*, disons `Infirmier`, attribué aux trois utilisateurs. C'est ce qu'illustre le tableau 5. On voit donc l'intérêt d'identifier des rôles : si un nouvel infirmier intègre l'organisation, il suffira de lui attribuer le rôle correspondant. Les administrateurs vont ainsi ajouter un seul tuple à la relation $\mathcal{URA}$. Dans la gestion classique basée sur les identités, on

	I	M	G	P	S
Alice	×	×			
Bob	×		×		
Charly	×			×	
Denise					×

	r1	w1	r2	w2	r3	w3	r4	w4	x4
Infirmier	×		×		×				
Médecin		×							
Gastrologue				×			×	×	×
Pédiatre						×	×	×	×
Secrétaire					×		×		

Tableau 5. *Affectations de rôles aux utilisateurs et de permissions aux rôles*

aurait dû définir explicitement trois tuples dans la relation $\mathcal{ACCESS}$: une pour chaque permission attribuée au rôle `Infirmier`.

4.1. *Définition de l'ingénierie des rôles*

Lors de l'introduction de la notion de rôle dans le contrôle d'accès on doit s'assurer que les utilisateurs disposent bien de tous les droits qu'ils avaient avant et *seulement* ceux-ci. L'objet de l'ingénierie de rôle est donc de décomposer une relation $\mathcal{ACCESS}$ existante en trois sous-relations telles[3] que :

$$\mathcal{URA} \bowtie \mathcal{RH} \bowtie \mathcal{PRA} = \mathcal{ACCESS}$$

Définition. *Ingénierie des rôles hiérarchisés Soit $\mathcal{ACCESS} \subseteq \mathcal{U} \times \mathcal{A} \times \mathcal{O}$ une matrice de contrôle d'accès existante. L'*ingénierie de rôles hiérarchisés *est l'activité qui consiste à identifier $\mathcal{URA} \bowtie \mathcal{RH} \bowtie \mathcal{PRA} = \mathcal{ACCESS}$ avec :*

– *un ensemble de rôles $\mathcal{R}$,*

– *une relation $\mathcal{URA} \subseteq \mathcal{U} \times \mathcal{R}$ et une relation $\mathcal{PRA} \subseteq \mathcal{R} \times \mathcal{A} \times \mathcal{O}$*

– *une relation $\mathcal{RH} \subseteq \mathcal{R} \times \mathcal{R}$*

- *réflexive* $\forall r \in \mathcal{R}, (r, r) \in \mathcal{RH}$,
- *transitive* $(r_1, r_2) \in \mathcal{RH}, (r_2, r_3) \in \mathcal{RH} \rightarrow (r_1, r_3) \in \mathcal{RH}$
- *antisymétrique* $(r_1, r_2) \in \mathcal{RH} \wedge (r_2, r_1) \in \mathcal{RH} \rightarrow r_1 = r_2$

Si l'on ne prend pas en compte la hiérarchie, c'est-à-dire si l'on s'intéresse seulement à trouver $\mathcal{URA}$ et $\mathcal{PRA}$ tels que $\mathcal{URA} \bowtie \mathcal{PRA} = \mathcal{ACCESS}$, l'ingénierie de rôles a pour but de représenter la relation $\mathcal{UP}$ du tableau 2 sous forme de deux relations binaires présentées dans le tableau 5. Ces deux dernières relations forment un exemple de résultat que l'on souhaite obtenir automatiquement. Dans cet exemple, cinq rôles (`I`,`M`,`G`,`P` et `S`) ont été identifiés pour structurer les droits du tableau 2.

La structure du treillis des concepts est beaucoup trop restrictive pour notre application. C'est une restriction importante portant sur la relation $\mathcal{RH}$ que l'on cherche à

3. L'opérateur $\bowtie$ désigne la jointure naturelle de deux relations.

identifier.. La structure de treillis impose que pour chaque couple de rôles $(r_1, r_2), \in \mathcal{R}^2$ il existe deux rôles $r_\top$ et $r_\bot$ tels que $r_\top \succeq r_1 \wedge r_\top \succeq r_2$ et $r_1 \succeq r_\bot \wedge r_2 \succeq r_\bot$.

4.2. *Sous-hiérarchie de Galois*

Avec une structure de treillis, on va multiplier les rôles dans le modèle de contrôle d'accès. De plus, ces rôles « imposés » n'ont pas nécessairement d'utilisateurs ou de permissions affectés. La *sous-hiérarchie de Galois* (SHG) a été introduite par Godin (Godin *et al.,* 1995) dans le cadre de la restructuration de hiérarchies de classes, pour réduire le nombre de concepts du treillis des concepts formels. En effet, dans le pire des cas on peut obtenir jusqu'à $2^{min(|O|,|A|)}$ concepts formels. Pour l'application au contrôle d'accès, une hiérarchie d'une telle taille n'est pas concevable.

Définition (Sous-hiérachie de Galois)**.** *Un concept C' est dit ancêtre (resp. descendant) d'un concept C ssi $C' <_C C$ (resp $C' >_C C$). Un objet (resp. un attribut) x est dit être* introduit *par un concept si x fait partie de l'extension (resp. de l'intention) de ce concept et qu'aucun ancêtre (resp. descendant) de ce concept ne contient x dans son extension (resp. intention). C'est le concept-objet (resp. concept-attribut) de x.*

La SHG *d'un treillis de Galois $(C, \leq_C)$ est le plus petit sous-ordre qui ne* contient que *les concepts-objets et les concepts-attributs du treillis de Galois.*

Les objets introduits par un concept forment son *extension réduite*, ses attributs son *intention réduite*. La sous-hiérarchie de Galois supprime donc les concepts qui ne sont *ni concepts objets ni concepts attributs*. C'est concepts peuvent être supprimés, car ils ne véhiculent pas d'information. Ils sont seulement imposés par la structure particulière de treillis. Pour un concept donné, l'extension réduite est l'ensemble des objets qu'il introduit explicitement, soit pour le contrôle d'accès ces utilisateurs *affectés*.

Pour le contrôle d'accès, cette propriété est particulièrement intéressante, car la SHG va supprimer de $(C, \leq_C)$ tous les rôles qui n'ont ni d'utilisateurs affectés, ni de permissions autorisées. En d'autres termes des rôles artificiels que très peu d'administrateurs auraient volontairement définis.

On ne garde ainsi que les *rôles-utilisateurs* (auxquels des utilisateurs sont affectés explicitement) et les *rôles-permissions* (auxquels des permissions sont affectées explicitement). Le choix de la SHG par rapport à la structure du treillis de concept de la FCA permet d'assurer les critères de correction, complétude et d'efficacité qui contribuent à former une « bonne hiérarchie » de rôles.

4.3. *Dérivation des autorisations*

L'objectif de tout modèle de contrôle d'accès est de permettre de dériver le triplet $\mathcal{ACCESS} \subseteq \mathcal{U} \times \mathcal{A} \times \mathcal{O}$ sur lequel prendre les décisions d'accès. Dans les modèles RBAC, il est interdit d'attribuer directement des permissions aux utilisateurs. En

d'autres termes, on ne peut pas manipuler la matrice 1 directement. On doit dériver $\mathcal{ACCESS}$ à partir des relations $\mathcal{URA}$, $\mathcal{RH}$ et $\mathcal{PRA}$.

Définition (Dérivation des autorisations dans les modèles RBAC)**.** *Les autorisations des utilisateurs sont dérivées à partir des affectations et de la hiérarchie de rôles :*

$$\forall (u, a, o) \in \mathcal{ACCESS} \Leftrightarrow$$
$$\exists r \in \mathcal{R}, u \in auth\_users(r) \wedge (a, o) \in auth\_perms(r)$$

Cette définition proposée est déduite à partir de (Ferraiolo *et al.,* 2003) où elle ne figure pas explicitement, du moins formellement. Nous proposons une *équivalence* entre le triplet $\mathcal{ACCESS}$ et la présence d'un rôle dans la politique RBAC. L'implication de droite à gauche exprime comment dériver les autorisations. En revanche, l'implication de gauche à droite signifie que l'affectation à un rôle est *obligatoire* pour obtenir une permission dans une politique RBAC (Ferraiolo *et al.,* 2003).

La question que nous nous posons est « qu'elle est la relation entre la SHG et le principe de dérivation des autorisations ? ». Nous remarquons que dans SHG, chaque utilisateur ou chaque permission n'apparait qu'une seule fois. L'extraction de la SHG dans la matrice de contrôle d'accès peut donc s'exprimer comme la recherche d'un ensemble partiellement ordonné tel que chaque utilisateur ne soit associé qu'à un seul rôle, et chaque permission à un seul rôle également.

$$\forall (u, a, o) \in \mathcal{ACCESS} \Leftrightarrow$$
$$\exists ! r \in R, u \in auth\_users(r) \wedge (a, o) \in auth\_perms(r)$$

Nous allons donc proposer une heuristique pour supprimer cette contrainte d'unicité du rôle qui n'a pas lieu d'être dans une politique RBAC. En effet, les affectations multiples (de rôles aux utilisateurs comme de permissions aux rôles) sont autorisées. Cette redondance de l'information répond à des besoins pratiques d'administration du contrôle d'accès (Crampton, 2003).

5. Automatisation de l'ingénierie des rôles

L'analyse des relations binaires pour la recherche des ensembles fermés, dont font partie les concepts formels, est un domaine de recherche actif, en France notamment (Pasquier *et al.,* 1999, Nourine *et al.,* 2002). Maintenant que nous avons établi le cadre pour l'ingénierie de rôle basé sur la FCA, nous allons pouvoir mettre à profit les résultats sur l'analyse des relations binaires.

5.1. *Choix d'un algorithme*

Les auteurs de (Arévalo *et al.,* 2007) ont proposé un algorithme PLUTON qui permet de calculer la SHG d'un contexte formel. Cet algorithme fait suite à (Berry *et al.,*

Description du critère	
FCA	RBAC
cardinalité de l'extension complète	utilisateurs autorisés
cardinalité de l'extension réduite	utilisateurs affectés
cardinalité de l'intention complète	permissions autorisés
cardinalité de l'intention réduite	permissions affectées
surface complète	utilisateurs $\times$ permissions autorisés
surface réduite	utilisateurs $\times$ permissions affectées
concepts immédiatement supérieurs	nombre de pères
concepts immédiatement inférieurs	nombre de fils

Tableau 6. *Critères principaux proposés pour le classement de concepts*

2005) qui propose une technique issue des graphes pour calculer une extension linéaire de la SHG. PLUTON est composé de trois sous-algorithmes TOMTHUMB, TOLINEXT et TOGSH :

– l'algorithme TOMTHUMB prend en entrée un contexte formel et retourne une liste ordonnée des labels[4] présents dans la SHG. Cette extension linéaire est un ordre total compatible avec l'ordre partiel $(C, \leq_C)$,

– l'algorithme TOLINEXT prend en entrée la liste résultat de TOMTHUMB et fusionne les paires extension et intention consécutives de cette liste qui appartiennent à un même concept. On obtient ainsi un tri topologogique de $(C, \leq_C)$,

– l'algorithme TOGSH calcule la SHG à partir du tri topologogique de TOLINEXT. Pour notre application, nous avons modifié cet algorithme pour qu'il calcule aussi les extensions et intentions complètes. Le surcoût engendré par ce calcul est négligeable face à la complexité de TOGSH (Arévalo *et al.,* 2007).

La complexité en temps théorique de TOMTHUMB est évaluée en $\mathcal{O}(|J|)$, l'implémentation brutale de TOLINEXT est décrite dans (Berry *et al.,* 2005) avec une complexité de $\mathcal{O}((|O| + |A|)^3)$, enfin TOGSH a été évalué en $\mathcal{O}((|O| + |A|)^2 * max(|O|, |A|)^2)$. D'autres algorithmes ont été proposés pour calculer la sous-hiérarchie de Galois, cependant PLUTON offre de bonnes performances vis-à-vis de ses concurrents (Arévalo *et al.,* 2007). La figure 3 (a) est la SHG retournée par TOGSH à partir du contexte formel du tableau 2.

5.2. *Étapes de la méthode proposée*

Nous proposons à l'expert du domaine chargé du chantier de d'ingénierie des rôles de mettre en œuvre notre approche semi-automatisée pour les assister dans leur travail.

4. Les labels sont des identifiants de lignes (ou de colonnes) identiques

En effet, comme plusieurs hiérarchies de concepts potentielles permettent de couvrir entièrement une relation binaire, le choix final ,et les éventuelles modifications à apporter au contexte formel si besoin, dépendent des besoins des organisations. Notre approche est décomposable en cinq étapes :

1) obtenir la *matrice des droits* d'accès du système sous forme de contexte formel,

2) calculer la *sous-hiérarchie de Galois* associée à l'aide de PLUTON, qui comporte les concepts potentiels du système.

Ces deux premières étapes forment le corps de l'automatisation. Les trois étapes suivantes seront itérées à partir du tri topologique de la SHG. L'intérêt est de proposer le processus d'élagage suivant comme un « calque » sur le tri topologogique des concepts formels. Cela permet à l'expert de faire varier ce calque en sélectionnant une mesure de pertinence. Nous pouvons ainsi garder en mémoire le tri topologique qui comporte toutes les unions et intersections pertinentes préalablement calculées.

3) *classer les concepts* obtenus selon une mesure de pertinence choisie. Le tableau 6 propose huit critères principaux selon lesquels classer les concepts,

4) *élaguer* la sous-hiérarchie afin de réduire sa taille en vue d'obtenir un diagramme de Hasse[5] représentable.

5) proposer cette hiérarchie de concepts à l'expert pour évaluation et analyse et itérer le processus pour raffiner la hiérarchie de rôles par élagages successifs.

La figure 3 (a) est la SHG obtenue à partir de la matrice du tableau 2. La figure 3 (b) est la hiérarchie obtenue après élagage en se limitant à six concepts. Le tri de pertinence est la taille de l'extension réduite. Le second tri pris en compte pour des concepts ayant même extension réduite est leur intention réduite.

Dans ces deux diagrammes, les six rôles potentiels sont représentés par des losanges et identifiés par les rôles du tableau 5. Les rôles identifiés par `X` et `Z` sont des concepts qui ne faisaient pas partie des cinq rôles initialement définis. À chaque rôle potentiel est également associée une note indiquant :

– le nombre d'utilisateurs affectés explicitement et implicitement,

– le nombre de permissions affectées explicitement et implicitement,

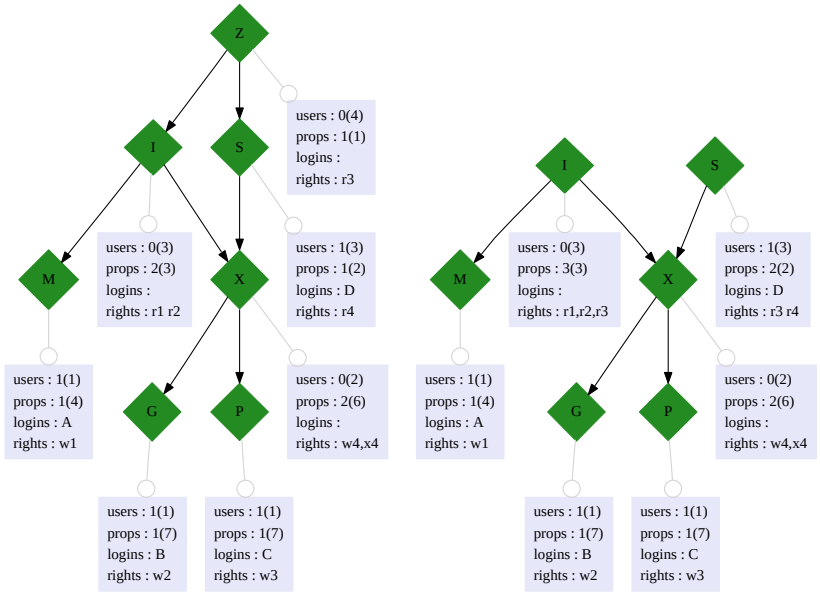
– la liste des utilisateurs affectés explicitement,

– la liste des permissions affectées explicitement,

5.3. *Obtention d'une relation binaire*

La première étape que nous avons identifiée dans la méthode est celle de l'obtention de la relation binaire à décomposer. Pour des raisons pratiques évidentes, les

5. Un diagramme de Hasse est une représentation visuelle d'un ordre fini qui en facilite la compréhension. Hormis les annotations, les illustrations de la figure 3 sont de tels diagrammes.



(a) Sous-hiérarchie de Galois (b) Hiérarchie élaguée

Figure 3. *Hiérarchies issues du contexte du tableau 2*

droits des utilisateurs ne sont pas directement gérés avec une telle matrice : il faut dériver et croiser des informations existantes sur les privilèges pour l'obtenir.

Le cas des systèmes utilisant des politiques à base de listes, comme les *Access Control Lists* ou les *Capability Lists* est facile à traiter : chaque liste représente soit une colonne soit une ligne de la matrice des droits. Cependant, il peut exister des dépendances *implicites* entre les permissions du système. Par exemple, il existe des relations implicites dans certains systèmes de gestion de bases de données Oracle. Donner le droit de *création* de table à un utilisateur lui attribue implicitement les autres droits de *manipulation de données* sur cette table. L'obtention d'une matrice nécessite donc une étape de traitement préalable dans le cas général.

5.4. *Classement des concepts*

Le nombre de concepts de la SHG peut être bien supérieur au nombre d'utilisateurs du système. Il faut donc réduire ce nombre en supprimant les concepts les moins pertinents comme rôles potentiels. Selon les organisations, l'expert peut privilégier de minimiser le *nombre de rôles* dans la hiérarchie, le nombre *d'affectations de permissions* aux rôles ou encore le nombre *d'affectations de rôles* aux utilisateurs.

Le tableau 6 présente huit critères utilisables pour classer les concepts selon leur pertinence. La notion de surface d'un concept, c'est-à-dire le nombre de croix qu'il couvre dans le contexte formel, est un indicateur pertinent. Une telle mesure est composée : c'est le produit de l'extension complète par l'intention complète. D'autres mesures peuvent être composées à partir de la liste de critères du tableau 6.

5.5. *Élagage de la sous-hiérarchie de Galois*

Une fois les concepts ordonnés, nous supprimons de la SHG les concepts considérés comme les moins pertinents. Comme nous voulons que notre approche soit correcte, il faut que la hiérarchie élaguée comporte *toute* l'information contenue dans le contexte formel d'origine. Le processus d'élagage va donc introduire des affectations *multiples* dans la hiérarchie afin de réduire le nombre de rôles. Le principe de l'algorithme de la figure 4 est le suivant :

1) calculer une matrice outil M de taille $|O| \times |A|$, qui stocke pour chaque couple $objet \times attribut$ le nombre de concepts où il est présent,

2) pour chaque concept $c = (O_c, A_c)$, du moins pertinent au plus pertinent, le supprimer s'il n'entraîne pas de perte d'information et mettre à jour M,

3) une fois la liste des concepts réduite, utiliser l'algorithme TOGSH pour reconstruire le graphe simplifié $G' = (V', E')$,

6. Évaluation de l'approche

6.1. *Approches ascendantes*

Les approches ascendantes permettent d'identifier des hiérarchies de rôles implicitement présents dans des permissions existantes. La hiérarchie obtenue peut être un arbre, une forêt d'arbre ou en ordre partiel général selon la proposition. Nous décrivons ici les principales approches d'identification automatisées de hiérarchies de rôles. En effet, en raison de leur nature, nous pouvons difficilement nous comparer aux méthodes descendantes.

Une première proposition nommée ROLEMINER est basée sur la suite d'outils IBM INTELLIGENT MINER (Kuhlmann *et al.,* 2003). La technique utilisée est celle du *demographic clustering*, hybride entre recherche de règles d'associations et classification non supervisée (*clustering*) (Grabmeier *et al.,* 2002). Les temps de calcul évoqués dans cette proposition sont de l'ordre de quelques heures pour environ 18.000 utilisateurs. Malheureusement, peu de détails sont donnés dans l'article et les algorithmes ne sont pas décrits, ce qui rend difficile la comparaison avec cette approche.

La proposition des auteurs de (Schlegelmilch *et al.,* 2005) est l'algorithme de clustering hiérarchique ORCA. La technique retenue est fondée sur les forêts d'arbres comme représentation des hiérarchies de rôles. Dans le cas général cependant, la hié-

Entrée : L : le tri topologique de la SHG
Sortie : $G' = (V', E')$: la SHG élaguée
begin
  `{initialisation}`
  soit M de taille $|O| \times |A|$: une matrice d'élagage
  for $c = (O_c, A_c) \in L$ **do**
    for $o \in O_c$ **do**
      for $a \in A_c$ **do**
        $M[o, a] \leftarrow M[o, a] + 1$
  `{boucle principale}`
  for $c = (O_c, A_c) \in L$ *du moins au plus pertinent* **do**
    for $o \in O_c$ **do**
      for $a \in A_c$ **do**
        if $M[o, a] > 1$ *(si c est supprimable)* **then**
          $M[o, a] \leftarrow M[o, a] - 1$
          $L \leftarrow L - c$
end
`{tous les concepts supprimables de` L `ont été supprimés}`
utiliser TOGSH sur L pour reconstruire le graphe $G' = (V', E')$

Figure 4. *Élagage de la* SHG *à partir d'une liste ordonnée*

rarchie est un ordre partiel sur les rôles, ce n'est pas nécessairement un arbre. Ceci conduit à une limitation importante d'ORCA : une permission ne peut être associée qu'à des rôles d'un même arbre. Un logiciel commercial *getRole* a suivi ces travaux. Les auteurs n'abordent pas les performances de leur approche ni la borne maximum du nombre de rôles que l'on peut obtenir.

Les auteurs de (Vaidya *et al.,* 2006) proposent d'utiliser l'énumération de sous-ensembles de permissions pour identifier des rôles. Les auteurs proposent deux algorithmes : une énumération complète, COMPLETEMINER, qui calcule tous les sous-ensembles de permissions possibles, et une énumération partielle FASTMINER, qui se limite au calcul d'intersections d'ensembles pris deux à deux. Les jeux d'essais proposés pour l'évaluation de leur approche sont des cas favorables à FASTMINER. L'algorithme COMPLETEMINER n'est pas évalué.

Les auteurs cette dernière proposition ont fait évolué leur approche (Vaidya *et al.,* 2007). Plutôt que de composer un algorithme d'extraction des rôles sur mesure, comme avec FASTMINER, ils mettent en relation le problème de l'ingénierie des rôles avec celui du *pavage d'une relation binaire* (Geerts *et al.,* 2004). La proposition ne présente pas de résultats expérimentaux. Nous pensons cependant qu'il existe des liens étroits entre leur cadre et le nôtre.

Jeu	Description	Paramètres
a	Nombre d'utilisateurs par rôle constant, nombre de permissions variable	`MRU=3` `NR=100, NU=2000`
b	Nombre de permissions par rôle constant, nombre de d'utilisateurs variable	`MRU=3, MPR=150` `NR=200, NP=1500`
c	Nombre de permissions constant, nombre d'utilisateurs par rôle variable	`MRU=3, MPR=150` `NP=1500`
d	Nombre d'utilisateurs, de permissions et de rôles constants, nombre de rôles par utilisateur variable	`NU=2500, NR=100` `MPR=50, NP=1500`

Tableau 7. *Paramètres utilisés pour les expérimentations*

6.2. *Méthodologie*

Les auteurs de (Vaidya *et al.,* 2006) ont proposé une méthodologie pour évaluer la pertinence et l'efficacité des approches ascendantes d'ingénierie de rôles. Nous l'avons utilisée pour évaluer la nôtre. Cette méthodologie est basée sur les principes du *reverse-engineering*. Elle consiste à générer aléatoirement une politique RBAC, à calculer la matrice des droits d'accès à partir de cette politique puis à vérifier si les rôles calculés à partir du contexte formel sont bien ceux initialement générés :

1) générer une politique RBAC avec ou sans hiérarchie des rôles,

2) construire la matrice des droits d'accès à partir de cette politique,

3) exécuter l'algorithme de découverte des rôles sur la matrice des droits d'accès,

4) classer les concepts calculés selon une mesure de pertinence sélectionnée,

5) parmi les n premiers concepts du classement, compter ceux qui sont effectivement des rôles générés dans la première étape. Le nombre de rôles tirés aléatoirement présents dans les n premiers identifiés donne une mesure de précision en $\%$.

6.3. *Résultats expérimentaux*

Nous avons réalisé quatre jeux d'essai synthétiques, **a**, **b**, **c** et **d** similaires à ceux réalisés dans (Vaidya *et al.,* 2006). Pour chaque jeu, quatre courbes sont proposées. Deux concernent la précision en $\%$ (sans hiérarchie : `P(f)`, avec : `P(h)`), les deux autres le temps d'exécution en secondes (sans hiérarchie : `T(f)`, avec : `T(h)`).

La figure 5 présente les résultats obtenus par notre proposition. Les expérimentations sont décrites dans le tableau 7. Le générateur de politique a comme paramètres le nombre de rôles `NR`, d'utilisateurs `NU`, de permissions `NP`, le nombre maximum de rôles `MRU` par utilisateurs ainsi que le nombre maximum de permissions par rôle `MPR`.

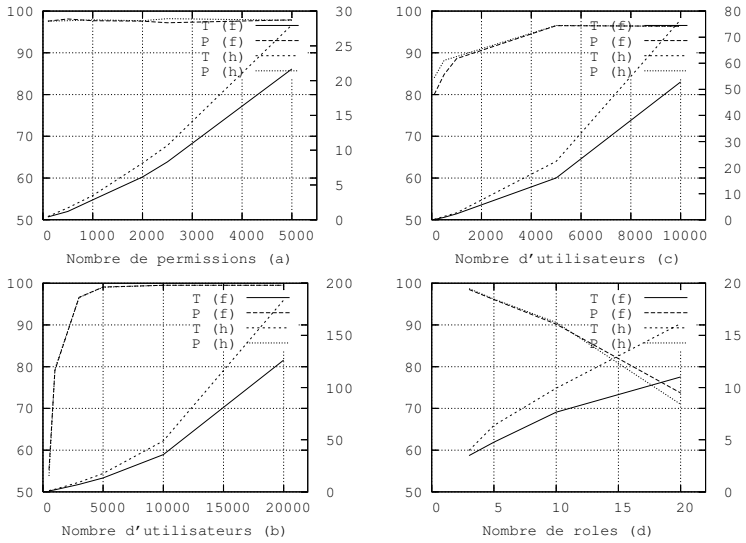


Figure 5. *Résultats expérimentaux, moyenne sur dix exécutions*

Le critère utilisé pour le classement des concepts est la taille de l'extension réduite. C'est le plus proche de celui choisi pour FASTMINER appelé *prioritization*.

Nous obtenons des mesures de précision légèrement supérieures à FASTMINER pour le jeu **a**, supérieures pour le jeu **b** et nettement supérieures pour le jeu **c**. Pour le jeu d'essai **d**, la précision décroît avec l'augmentation du nombre de rôles attribués à chaque utilisateur. Les rôles générés ne sont pas les premiers du classement des concepts, mais sont cependant majoritairement présents dans la SHG : la mesure de pertinence par taille de l'extension réduite n'est donc pas adaptée à ce jeu d'essai.

Notre approche est entre 50 et 100 fois plus rapide que les performances de FASTMINER présentées dans (Vaidya *et al.,* 2006). Même si le choix du langage et la programmation influent sur les performances, la différence est suffisamment significative. Les résultats avec un générateur de politique hiérarchisé ou non sont assez similaires, le calcul est sensiblement plus coûteux dans le cas des hiérarchies, car le contexte formel est plus dense.

Nous avons également évalué les performances sur les jeux d'essai proposés par la FIMI (*Frequent Itemset Mining Implementations*), qui met à disposition plusieurs jeux d'essai volumineux pour comparer les algorithmes de data-mining. Par exemple, pour le jeu `connect` de taille 67557 objets et 129 attributs, la sous-hiérarchie de Galois est calculée en un peu moins de 30 minutes.

6.4. *Jeux d'essais réels*

Nous avons eu à notre disposition deux matrices des droits issues d'une même organisation : un établissement hospitalier. L'évaluation de la pertinence des résultats obtenus sur ces jeux d'essais est relativement difficile. En effet, nous ne disposons ni de la hiérarchie de rôles *a priori* comme pour les jeux d'essais synthétiques, ni d'un expert qui puisse évalue nos résultats.

6.4.1. *Contrôle d'accès à une application spécifique*

Ce jeu d'essai est une matrice des droits d'accès d'une des applications de l'établissement. Il concerne 37 utilisateurs et les 184 permissions spécifiques à cette application. La SHG calculée à partir de ce jeu d'essai comporte 33 concepts et 117 relations hiérarchiques. Le cas limite pour cet exemple est une politique de 8 rôles avec 2 relations d'héritage. Le temps de calcul pour ce jeu d'essai est inférieur à 0.05 seconde.

6.4.2. *Contrôle d'accès à des applications*

L'autre jeu d'essai est la matrice d'accès aux applications de l'établissement, qui indique parmi les 12 applications existantes qu'elles sont celles autorisées pour chacun des 1582 utilisateurs de l'organisation. La SHG calculée à partir de ce jeu d'essai comporte 48 concepts et 93 relations hiérarchiques. L'élagage maximum conduit à une politique de 12 concepts sans hiérarchie : il s'agit du cas limite où chaque rôle donne exactement une permission. Le temps de calcul pour ce jeu d'essai est inférieur à 0.1 seconde.

La SHG est difficile à représenter graphiquement. Le choix d'une mesure de pertinence et du nombre de rôles minimum que l'on souhaite permet de calculer des politiques de tailles variables, comprises entre 12 et 8 rôles. Parmi les 12 applications de la matrice, deux d'entre elles ne sont utilisées que par exactement *une seule* personne chacune. Cette attribution de droit crée ainsi deux concepts propres à ces utilisateurs, qui *ne peuvent pas être supprimés*.

Pour ce jeu d'essai, un expert serait peut-être en mesure de faire une analyse croisée des concepts de la SHG entre deux aspects des rôles :

– les services de santé qui composent l'établissement,

– les fonctions des utilisateurs au sein de l'établissement,

6.5. *Catégorisation de rôles obtenus*

D'après les définitions de la SHG, on peut obtenir trois grandes catégories de rôles, selon qu'un concept introduise des objets, des attributs ou les deux. On peut interpréter ces trois catégories comme :

– les rôles *pertinents* : il s'agit des concepts à la fois concepts-attributs et concepts-objets. Ces rôles sont significatifs : des permissions et des utilisateurs leur sont explicitement affectés.

– les rôles *abstraits* : il s'agit des concepts-attributs : des rôles auxquels *aucun utilisateur n'est affecté directement*, les utilisateurs étant affectés soit à leur ancêtres, soit à leurs descendants dans la hiérarchie. Il est intéressant d'en garder certains pour leur *pouvoir structurant*,

– les rôles *spécifiques* à un groupe d'utilisateurs : il s'agit des concepts-objets, ces rôles sont à éviter. Il vaut mieux privilégier l'affectation multiple plutôt que d'avoir à maintenir *un rôle spécifique* à un groupe d'utilisateur qui n'apporte *aucune permission supplémentaire* que celles de ses ancêtres.

7. Conclusion

7.1. *Bénéfices de l'approche*

Pour l'application à la découverte d'une hiérarchie de rôles potentielle, la structure de sous-hiérarchie de Galois extraite avec l'algorithme PLUTON est adéquate, car :

– la notion de *concept* peut être une caractérisation formelle de la notion de rôle : une abstraction définissable par ses affectations d'utilisateurs et de permissions,

– les concepts sont des *regroupements maximaux* d'objets et d'attributs, ce qui répond à un besoin en terme d'ingénierie : les rôles doivent comporter le maximum d'utilisateurs et de permissions,

– la FCA bénéficie d'une *caractérisation formelle*. De plus, la SHG lève la restriction de treillis et forme ainsi une structure utilisable sur laquelle on effectuera des traitements *a posteriori*,

– PLUTON offre des performances suffisantes pour le mettre en œuvre sur des politiques considérables : dizaines de milliers d'utilisateurs et de permissions.

Notons que dans le cas général, la relation à décomposer peut être une autre relation que celle entre utilisateur et permission. En effet, il existe des modèles de contrôle d'accès qui comprennent plusieurs intermédiaires entre utilisateurs et permissions, comme celui introduit par Epstein représenté par la figure 2.

7.2. *Sémantique de la relation d'héritage*

Nous avons repris la formalisation de RBAC de (Ferraiolo *et al.,* 2003). Or, il a été remarqué qu'elle est ambiguë, voire erronée (Li *et al.,* 2007). La relation d'héritage peut servir soit à regrouper des utilisateurs, des permissions ou les deux.

> ***Suggestion 5*** *The semantics of role inheritance should be clearly specified and discussed.*

L'étude et la mise en perspective entre contrôle d'accès et analyse de concepts formels nous a conduits à identifier des liens entre les utilisateurs autorisés d'un rôle et ses permissions autorisées. Nous avons principalement considéré les rôles comme des regroupements d'utilisateurs *et* de permissions. L'outillage théorique issu de la FCA pourrait permettre de redéfinir la notion de hiérarchie de rôles comme la suivante :

Définition (Sémantique de la relation d'héritage)**.** $r_1 \succeq r_2$ *si seulement si toutes les permissions autorisées à* r_2 *sont autorisées à* r_1 *et tous les utilisateurs autorisés de* r_1 *sont autorisés de* r_2 *:*

$$r_1 \succeq r_2 \Leftrightarrow auth\_perms(r_2) \subseteq auth\_perms(r_1) \wedge \\ auth\_users(r_1) \subseteq auth\_users(r_2)$$

7.3. *Heuristique de sélection*

La méthode d'élagage que nous avons proposée relève de l'heuristique : nous proposons à des experts de sélectionner l'indicateur de pertinence qui convient le mieux à leur objectif de modélisation. La mesure de la complexité des politiques RBAC a été abordée dans (Jaeger, 2001). On pourrait utiliser cette mesure comme critère de tri pour les concepts à sélectionner comme pertinent. L'heuristique de sélection consisterait alors à supprimer les concepts qui compliquent trop la politique. On peut également envisage d'utiliser la mesure de complexité d'une politique pour comparer la qualité des propositions d'ingénierie des rôles. On comparerait alors les résultats des différentes méthodes selon la complexité de la politique obtenue.

7.4. *Lien avec le pavage*

La recherche de rectangles dans une relation binaire est un socle commun sur lequel ont été définis plusieurs problèmes différents de fouille de données (Geerts *et al.,* 2004). Les auteurs de (Vaidya *et al.,* 2006) ont défini le problème de l'ingénierie des rôles comme celui du pavage d'une relation binaire. Leur méthode d'analyse et d'interprétation consiste donc à identifier un ensemble de rôles le plus petit possible, puis d'éventuellement modifier itérativement cet ensemble pour obtenir une hiérarchie satisfaisante. Notre définition de l'ingénierie des rôles est différente. Elle est exprimable comme la recherche d'un ensemble partiellement ordonné dans une relation binaire. De plus, nous nous basons sur la SHG pour itérer le processus de raffinement de l'expert. Comme les deux approches partent d'une relation binaire, des ponts peuvent exister entre les deux définitions. Le pavage pourrait être une caractérisation d'un élagage maximal de la SHG. Le rôle de l'expert serait alors de trouver la hiérarchie la plus satisfaisante entre ces deux extrêmes.

7.5. *Atelier d'ingénierie*

Parmi les concepts présents dans la SHG, certains sont *sup-irréductibles* ou *inf-irréductibles* (Ganter *et al.,* 1997) : ils ne peuvent pas être exprimés comme l'union ou l'intersection d'autres concepts. Nous pensons qu'il serait intéressant d'introduire cette notion dans notre proposition : à partir d'une caractérisation formelle, nous apposerions des *labels* sur les concepts de la SHG. On pourrait faire intervenir ces labels accompagnés d'autres indicateurs sur les notes que nous avons faits figurés sur les hiérarchies de la figure 2. L'objectif serait de faciliter la sélection des concepts pertinents et l'interprétation de la hiérarchie de rôles obtenue par l'expert.

8. Bibliographie

Arévalo G., Berry A., Huchard M., Perrot G., Sigayret A., « Performances of Galois Subhierarchy-building Algorithms », *5th International Conference on Formal Concept Analysis, ICFCA'07*, vol. 4390 of *LNCS*, 2007.

Berry A., Huchard M., McConnell R. M., Sigayret A., Spinrad J., « Efficiently Computing a Linear Extension of the Sub-hierarchy of a Concept Lattice. », *3rd International Conference on Formal Concept Analysis, ICFCA'2005*, vol. 3403 of *LNCS*, p. 208-222, 2005.

CERT/CC, Service U. S., magazine C., E-CrimeWatch Survey, Technical report, 2005.

Coyne E. J., « Role engineering », *RBAC '95 : Proceedings of the first ACM Workshop on Role-based access control*, ACM Press, New York, NY, USA, p. 4, 1996.

Crampton J., « Specifying and enforcing constraints in role-based access control », *SACMAT*, Association for Computing Machinery, ACM Press, New York, NY, USA, p. 43-50, 2003.

Epstein P. E., Engineering of Role/Permissions assignments, PhD thesis, Fairfax University, VA, USA, 2002.

Ferraiolo D. F., Kuhn R. D., Chandramouli R., *Role-Based Access Control*, Artech House Publishers, 2003.

Gallaher M. P., O'Connor A. C., Kropp B., The Economic Impact of Role-Based Access Control, Technical report, Planning report 02-1, National Institute of Standards and Technology, 2002.

Ganter B., Wille R., *Formal Concept Analysis : Mathematical Foundations*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997. Translator-C. Franzke.

Geerts F., Goethals B., Mielikäinen T., « Tiling Databases », *in* , E. Suzuki, , S. Arikawa (eds), *Discovery Science*, vol. 3245 of *Lecture Notes in Computer Science*, Springer, p. 278-289, 2004.

Godin R., Mineau G., Missaoui R., Mili H., « Méthodes de Classification Conceptuelle Basées sur les Treillis de Galois et Applications », *Revue d'Intelligence Artificielle*, vol. 9, n° 2, p. 105-137, 1995.

Grabmeier J., Rudolph A., « Techniques of Cluster Algorithms in Data Mining », *Data Min. Knowl. Discov.*, vol. 6, n° 4, p. 303-360, 2002.

Jaeger T., « Managing access control complexity using metrices », *SACMAT 2001, 6th ACM Symposium on Access Control Models and Technologies, 2001, Proceedings*, ACM Press, New York, NY, USA, p. 131-139, 2001.

Kuhlmann M., Shohat D., Schimpf G., « Role mining - revealing business roles for security administration using data mining technology », *SACMAT*, Association for Computing Machinery, ACM Press, p. 179-186, 2003.

Lampson B. W., « Protection », *SIGOPS Oper. Syst. Rev.*, vol. 8, n° 1, p. 18-24, 1974.

Li N., Byun J.-W., Bertino E., « A Critique of the ANSI Standard on Role-Based Access Control », *IEEE Security and Privacy*, vol. 5, n° 6, p. 41-49, 2007.

Neumann G., Strembeck M., « A scenario-driven role engineering process for functional RBAC roles », *SACMAT 2002, 7th ACM Symposium on Access Control Models and Technologies, Proceedings*, ACM Press, New York, NY, USA, p. 33-42, 2002.

Nourine L., Raynaud O., « A fast incremental algorithm for building lattices. », *J. Exp. Theor. Artif. Intell.*, vol. 14, n° 2-3, p. 217-227, 2002.

Pasquier N., Bastide Y., Taouil R., Lakhal L., « Discovering Frequent Closed Itemsets for Association Rules. », *in* , C. Beeri, , P. Buneman (eds), *ICDT*, vol. 1540 of *LNCS*, Springer, p. 398-416, 1999.

Roeckle H., Schimpf G., Weidinger R., « Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. », *ACM Workshop on Role-Based Access Control*, p. 103-110, 2000.

Sandhu R. S., Coyne E. J., Feinstein H. L., Youman C., « Role-Based Access Control Models. », *IEEE Computer*, vol. 29, n° 2, p. 38-47, 1996.

Schlegelmilch J., Steffens U., « Role mining with ORCA », *in* , E. Ferrari, , G.-J. Ahn (eds), *SACMAT 2005, 10th ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden, June 1-3, 2005, Proceedings*, ACM Press, New York, NY, USA, p. 168-176, 2005.

Vaidya J., Atluri V., Guo Q., « The role mining problem : finding a minimal descriptive set of roles », *in* , V. Lotz, , B. M. Thuraisingham (eds), *SACMAT*, ACM, p. 175-184, 2007.

Vaidya J., Atluri V., Warner J., « RoleMiner : mining roles using subset enumeration », *13th ACM conference on Computer and Communications Security, CCS'06*, ACM Press, New York, NY, USA, p. 144-153, 2006.

Wille R., « Conceptual Graphs and Formal Concept Analysis », *5th International Conference on Conceptual Structures, ICCS'97*, p. 290-303, 1997.