République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderahmane Mira de Béjaïa

Faculté des Sciences Exactes

Département d'Informatique

École Doctorale Réseaux et Systèmes Distribués

Mémoire de Magistère

En Informatique

Option

Réseaux et Systèmes Distribués

Thème

Spécification d'un Modèle Formel pour l'Expression des Autorisations d'Accès aux Web Services

---

Présenté par

DJEBARI Nabil

Devant le jury composé de :

Président	Mohammed Said RADJEF	Professeur	U. Béjaia
Rapporteur	Djamil AISSANI	Professeur	U. Béjaia
Examinateur	Zizette BOUFAIDA	Professeur	U. Constantine
Examinateur	Karima BENATCHBA	Professeur	E.S.I
Invitée	Hassina NACER eps TALANTIKITE	MCB	U. Béjaia.

Promotion 2009-2010

Dédicaces

À la mémoire de mon père
À ma mère,
À mes frères et mes s œurs,
À toute ma famille,
À mes amis et collègues.

Remerciements

Au terme de ce travail, je remercie le BON DIEU tout puissant qui m'a donné la force et la volonté d'achever ce travail et nous lui rendons grâce.

Aucune œuvre humaine ne peut se réaliser sans la contribution d'autrui. Ce mémoire est le résultat d'un effort constant, cet effort n'aurait pu aboutir sans la contribution d'un nombre de personnes. Ainsi se présente l'occasion de les remercier : Mes remerciements vont à Monsieur Djamil AISSANI, Professeur à l'université de Bejaia, mon Directeur de Mémoire pour avoir accepté de m'encadrer et pour m'avoir témoigné sa confiance totale.

Je remercie mon Co-promotrice, Mme Nacer Talantikite Hassina, maitre de conférences à l'université de Bejaïa pour la qualité de ses conseils, sa disponibilité ainsi que le degré de responsabilisation de son encadrement. Je la remercie d'avoir consacré beaucoup de son temps pour les nombreuses relectures de mon document.

J'exprime ma profonde gratitude à Monsieur Mohamed Said RADJEF, professeur à l'université de Béjaia pour l'honneur qu'il me fait en présidant mon jury, ainsi qu'à madame Zizette BOUFAIDA, Professeur à l'Université de Constantine et à Madame Karima BENATCHBA, professeure à l'école supérieure d'informatique, d'avoir accepté de juger mon travail.

Je remercie vivement les enseignants et les responsables de l'École Doctorale d'Informatique de Béjaia pour tous les efforts qu'ils ont fournis pour la réussite et l'épanouissement de l'école.

Je souhaite également saluer tous mes amis et camarades en postgraduation.

Je remercie également mademoiselle Anissa HADDADI chef de service de la scolarité à la faculté SEGC, pour son aide, son soutient morale et sa contribution à la réalisation de ce travail.

Je remercie de tout cœur toute ma famille pour la confiance, le soutien et l'aide qu'ils m'ont apportée durant toute la durée de ce mémoire.

A tous ceux qui d'une façon ou d'une autre, de près ou de loin, m'ont apporté leurs contributions et leur soutien dans l'élaboration de ce travail ; qu'ils trouvent ici l'expression de ma profonde gratitude.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

La collaboration entre les différentes organisations est devenue aujourd'hui un besoin pressant. En effet, l'accès aux systèmes d'information s'appuie de plus en plus sur des technologies Internet. Les efforts de standardisation dans ce contexte ont accentué l'engouement des personnes et des organisations (aussi bien académiques, qu'industrielles, commerciales, ou institutionnelles) pour l'utilisation de l'Internet. Ainsi, distribués, des systèmes hétérogènes deviennent très courants, comme les organisations intègrent des applications s'exécutant sur des plateformes différentes. En conséquence, il y a une demande croissante pour les architectures et les technologies qui supportent la connexion et le partage des ressources flexibles grâce à l'utilisation de protocoles standardisés. Néanmoins, les solutions disponibles à cet effet ne garantissent pas une parfaite interopérabilité, elles sont complexes, cas de CORBA (Common Object Request Broker Architecture), ou pas flexibles, cas de RMI (Remote Method Invocation) et DCOM (Distributed Component Object Model).

Les services Web ont provoqué une forte évolution dans le monde de l'informatique distribuée, et un bouleversement majeur dans la façon de concevoir des architectures. Un des intérêts du service Web est de faciliter l'interconnexion entre les différentes applications distantes, indépendamment des plateformes et des langages de programmation utilisés. Les services Web semblent être la solution de l'avenir pour implémenter les systèmes distribués. Aujourd'hui, ces services sont distribués à large échelle sur Internet. Le développement d'Internet, la structuration des données via XML, et la recherche d'interopérabilité sont autant de facteurs qui ont favorisé l'essor des services Web. Les services Web constituent la technologie de base pour le développement d'architectures orientées servies. Ces derniers sont des modèles qui définissent un système par un ensemble de services logiciels distribués, qui fonctionnent indépendamment les uns des autres afin de réaliser une fonctionnalité globale.

Le paradigme des services Web a suscité beaucoup d'intérêt de la part de la communauté académique et industrielle, ce qui a permis de donner naissance à de nouveaux axes de

recherche telle que la sécurité des services Web.

La distribution et le déploiement des services sur des plates-formes hétérogènes, ouvrent des failles de sécurité importantes. En effet, il existe des risques d'écoute, de vol et de divulgation des informations qui circulent entre les différents acteurs. Les axes de sécurité les plus importants pour des applications basées sur les services Web sont l'authentification, la sécurité des messages transmis, et le contrôle d'accès.

Un modèle de contrôle d'accès protège les opérations des services Web contre les utilisateurs qui ne répondent pas aux exigences d'autorisation du fournisseur de service Web. L'application de contrôle d'accès implique : trouver des solutions pour représenter correctement l'identité des utilisateurs de service (identification), et décider si des utilisateurs du service sont autorisés à utiliser les services Web.

A ce jour, aucun standard n'a été proposé pour les modèles de contrôle d'accès pour les services Web, plus particulièrement les services Web composites. La majorité des modèles de contrôle d'accès proposés dans la littérature ont été développés pour une politique spécifique. Bien que ces modèles couvrent différentes situations et des types de données d'un service Web donné, ils paraissent complexes, voire impossibles pour d'autres.

Dans ce mémoire, nous nous intéressons au contrôle d'accès aux services Web et à la spécification formelle du modèle de contrôle d'accès. Notre objectif est de proposer un modèle de contrôle d'accès aux services Web composites décrit sémantiquement et de représenter formellement le modèle proposé.

Afin de mieux renforcer la sécurité des services Web, nous avons proposé un modèle de contrôle d'accès distribué, où chaque service Web contrôle l'accès à ses ressources, et afin de réduire la complexité en hétérogénéité engendrée par la diversité des politiques de contrôle d'accès, nous avons regroupé les services Web en cercles de confiances où chaque cercle de confiance regroupe les services Web qui utilisent la même politique de contrôle d'accès, et une ontologie de domaine afin de masquer l'hétérogénéité sémantique.

Notre mémoire est structuré en cinq chapitres :

- **Chapitre 1 :** Il est consacré à l'état de l'art des services Web, les standards de base des services Web, le Web sémantique et la composition des services Web ;
- **Chapitre 2 :** Il introduit les notions de base du contrôle d'accès, les modèles de contrôle d'accès aux systèmes d'information et les modèles de contrôle d'accès aux services Web ;
- **Chapitre 3 :** Il est consacré aux modèles formels pour les politiques de contrôle d'accès ainsi qu'une comparaison de ces derniers ;

- **Chapitre 4 :** Il décrit le modèle proposé de contrôle d'accès aux services Web composites ainsi que sa modélisation par les graphes conceptuels ;
- **Chapitre 5 :** Il présente une mise en œuvre d'un prototype du modèle de contrôle d'accès proposé, ainsi que quelques scénarios d'exécution.

Enfin, ce rapport s'achève par une conclusion générale dans laquelle nous présentons un rappel des principaux résultats obtenus et quelques perspectives.

Une cartographie de notre travail est illustrée par la figure 1.

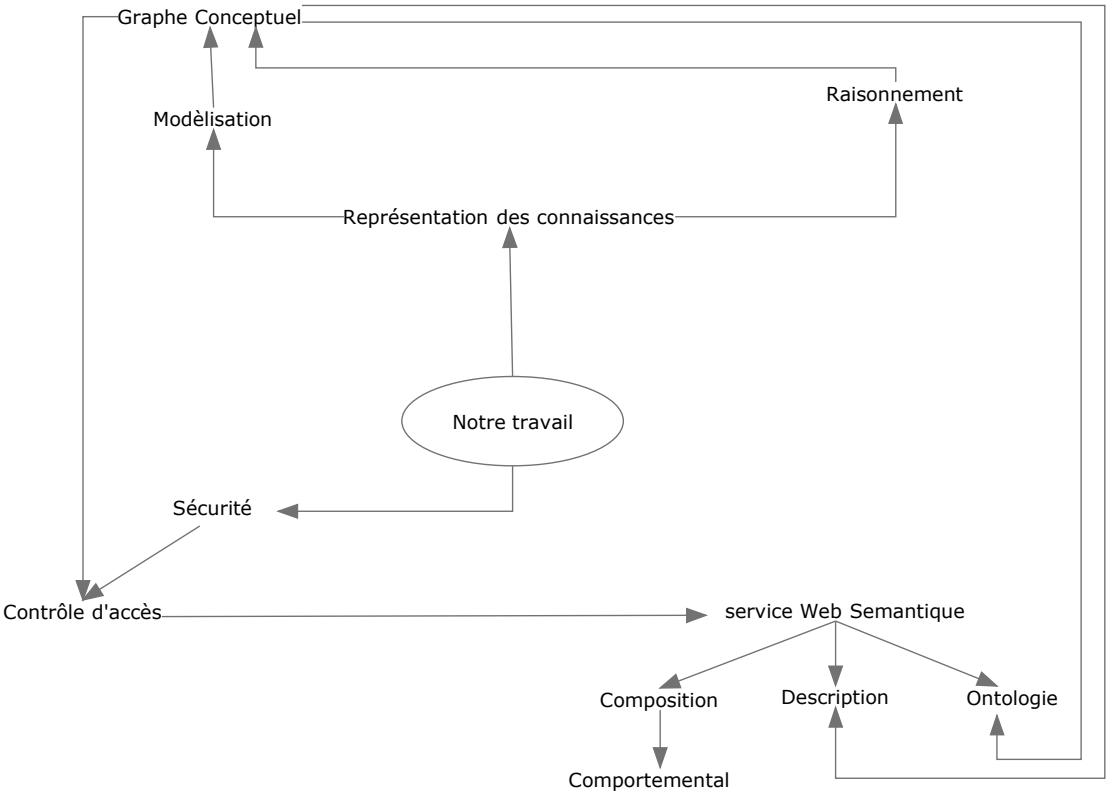

FIGURE 1 – Positionnement de notre travail

Chapitre 1

Services Web

1.1 Introduction

Le développement rapide des systèmes distribués a entrainé l'apparition de multiples protocoles tels que : DCOM [1] [33] , RMI [2] [33] , CORBA [3] [32], ...etc. cette diversité de technologies rend les interactions entre applications plus compliqué. La technologie des services Web offre de fortes potentialités pour surmonter les problèmes d'interopérabilité des systèmes. Elle constitue un cadre prometteur pour l'intégration des applications, et pour la gestion des interactions entre divers partenaires dans un environnement distribué, hétérogène, ouvert et versatile qui est le Web.

L'objectif ultime de l'approche services Web est de transformer le Web en un dispositif distribué de calcul, où les services Web peuvent interagir de manière intelligente en étant capables de se découvrir automatiquement, de négocier entre eux et de se composer en des services plus complexes [17].

1.2 Définition des services Web

Les services Web sont vus comme des applications accessibles à d'autres applications sur le Web. De nos jours, le terme service Web est souvent utilisé, mais pas toujours avec la même signification, car la signification dépend de l'application de cette technologie. Il existe plusieurs définitions pour les services Web parmi elles :

Définition 1 : Le consortium W3C [4] définit un service Web comme étant : « *une application, ou un composant logiciel qui vérifie les propriétés suivantes* » [25] :

1. Distributed Component Object Model
2. Remote method invocation
3. Common Object Request Broker Architecture
4. The World Wide Web Consortium, www.w3.org : consortium chargé de promouvoir la compatibilité des technologies du World Wide Web.

- Il est identifié par un URI [5] ;
- Ses interfaces et ses liens peuvent être décrits en XML [6] [81] ;
- Sa définition peut être découverte par d'autres services Web ;
- Il peut interagir directement avec d'autres services Web à travers le langage XML en utilisant des protocoles Internet standards.

Définition 2 : « *Les services Web sont des applications autodescriptives, modulaires et faiblement couplées qui fournissent un modèle de programmation et de déploiement d'applications, basé sur des normes, et s'exécutant au travers de l'infrastructure Web* » [50].

Définition 3 : « *Un service Web est une application accessible à partir du Web. Il utilise les protocoles Internet pour communiquer, et utilise un langage standard pour décrire son interface.* » [59].

Définition 4 : « *Un service Web est un système logiciel identifié par un URI, dont les interfaces publiques et les incarnations sont définies et décrites en XML. Sa définition peut être découverte dynamiquement par d'autres systèmes logiciels. Ces autres systèmes peuvent ensuite interagir avec le service Web d'une façon décrite par sa définition, en utilisant des messages XML transportés par des protocoles Internet.* » [37].

Il existe probablement autant de définitions de services Web que d'entreprises qui les créent, mais presque toutes ces définitions s'accordent à satisfaire les points suivants :

- Les services Web proposent aux utilisateurs du Web des fonctionnalités pratiques grâce à un protocole Web standard ;
- Les services Web offrent un moyen de décrire leurs interfaces suffisamment en détail pour permettre à un utilisateur de créer une application cliente capable de converser avec eux. Cette description est généralement fournie dans un document XML ;
- Les services Web sont publiés afin que les utilisateurs potentiels puissent les trouver facilement ;
- Un service Web ne doit pas être lié à un système d'exploitation ou à un langage de programmation.

1.3 Caractéristiques des services Web

Les services Web ont les caractéristiques suivantes :

5. Uniform Resource Identifier
6. Extensible Markup Language

Accessible via un URI : Un service Web est accessible en spécifiant son URI, c'est-à-dire que le service Web est caractérisé par un seul objet et une seule fonctionnalité, mais on peut faire la construction d'une application logicielle très large comportant plusieurs fonctionnalités, afin de sélectionner les fonctionnalités qui sont recherchées par leurs URI spécifiques.

Accessibilité universelle : Un service Web peut être défini, décrit et découvert à travers le Web qui facilite leurs accessibilités. Non seulement les utilisateurs des services Web peuvent localiser les services appropriés, mais aussi les services Web peuvent se décrire et se publier par eux-mêmes pour qu'il soit possible de se lier et d'interagir les uns avec les autres.

Autocontenu et autodescriptif : Un service Web contient dans sa description toutes les informations nécessaires à l'utilisation des ses applications, sous la forme de trois fonctions : Trouver, Décrire et Exécuter.

Modularité : Les services Web fonctionnent de manière modulaire et non pas intégrée. Cela signifie qu'au lieu d'intégrer dans une seule application globale toutes les fonctionnalités, on crée (ou on récupère) plusieurs applications spécifiques qu'on fait intéropérées entre elles, et qui remplissent chacune une de ces fonctionnalités.

Composante faiblement couplée : Cette caractéristique permet à un service Web de localiser et de communiquer avec les autres services Web dynamiquement pendant l'exécution.

Basè sur XML : XML constitue la technologie de base des architectures des services Web, il est un facteur important pour contourner les barrières techniques.

1.3.1 Architecture de référence

L'architecture de référence représente une vue globale sur les rôles ainsi que les relations qui les unissent afin de montrer le fonctionnement global des services Web. Chaque rôle dans l'architecture est défini par l'ensemble des relations qu'il mène avec les autres rôles. L'architecture donne également des indications sur le cycle de développement des services Web ainsi que leur cycle de vie [22].

Cette architecture vise trois objectifs importants [42] :

- Identification des composants fonctionnels ;
- Définition des relations entre ces composants ;

– Établissement d'un ensemble de contraintes sur chaque composant de manière à garantir les propriétés globales de l'architecture.

L'architecture de référence s'articule autour des trois rôles suivants :

Le fournisseur de service : Correspond au propriétaire du service Web. D'un point de vue technique, il est constitué par la plateforme d'accueil du service Web ;

Le client : Correspond au demandeur de service Web. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service Web. L'application cliente peut être elle-même un service Web ;

L'annuaire des services : Correspond à un registre de descriptions des services Web offrant des facilités de publication des services Web à l'intention des fournisseurs, ainsi que des facilités de recherche des services Web.

Les interactions de base entre ces trois rôles incluent les opérations de publication, de recherche et de liens d'opérations. La figure 1.1 représente les différents concepts et relations de l'architecture de référence des services Web. Nous citons, notamment les standards émergents suivants :

– **SOAP** [7] **:** Définit un protocole de transmission de messages basé sur XML ;
– **WSDL** [8] **:** Introduit une grammaire pour la description des services Web ;
– **UDDI** [9] **:** Fournit l'infrastructure de base pour la publication, et la découverte des services Web.

Le cycle de vie d'un service Web est :

1. Le fournisseur de services définit la description de son service et publie sa description WSDL dans un annuaire de service UDDI ;
2. Le client utilise les facilités de recherche disponibles au niveau de l'annuaire UDDI pour retrouver et sélectionner un service donné ;
3. Le client examine ensuite la description du service sélectionné pour récupérer les informations nécessaires lui permettant de se connecter au fournisseur du service et d'interagir avec l'implémentation du service considéré.

7. Simple Object Access Protocol : http ://www.w3.org/TR/soap/
8. Web Services Description Language : http : //www.w3.org/TR/wsdl/
9. Universal Description Discovery and Integration : http ://www.uddi.org/ par OASIS Open pour UDDI

Cependant, cette infrastructure n'est pas suffisante pour permettre une utilisation effective des services Web, dans les domaines dont les exigences vont au-delà de la capacité d'interactions simples via des protocoles standards. Par exemple, dans le domaine du e-business.

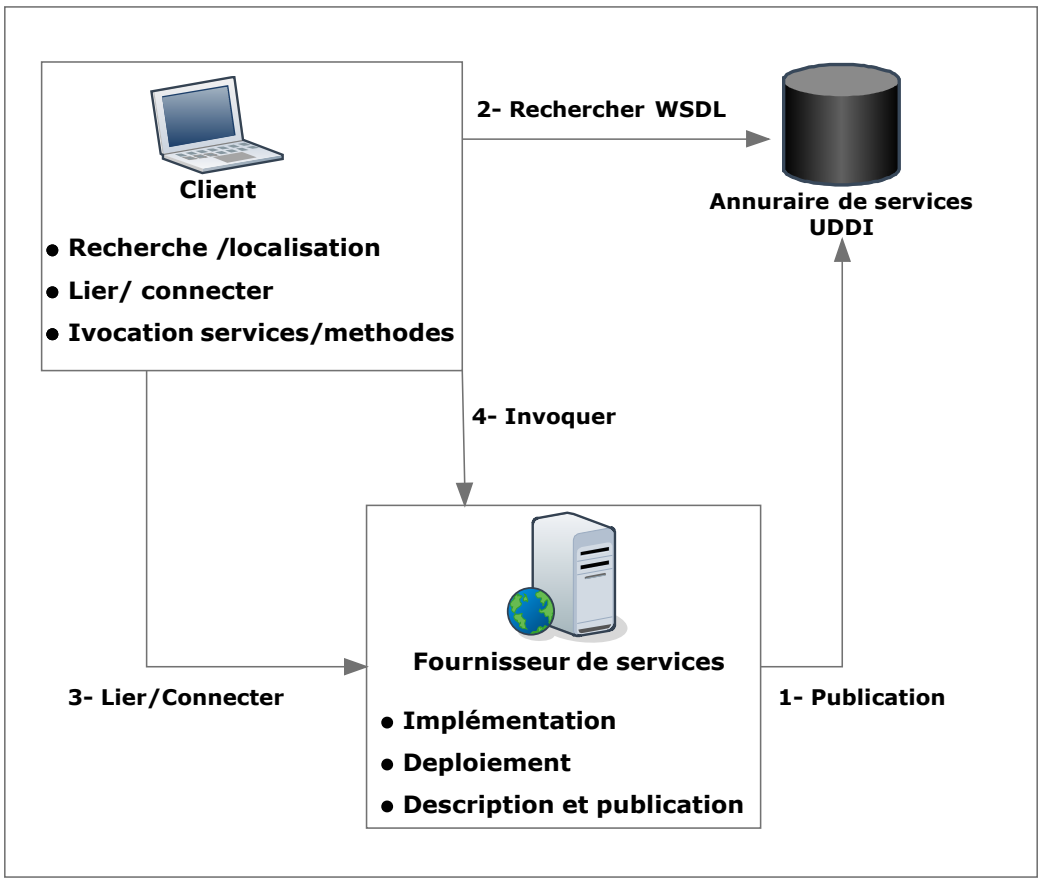

FIGURE 1.1 – Architecture de référence

1.3.2 Architecture étendue :

Une architecture étendue est constituée de plusieurs couches se superposant les unes sur les autres, d'où le nom de pile des services Web. La pile est constituée de plusieurs couches, chaque couche s'appuyant sur un standard particulier. On retrouve, au-dessus de la couche de transport, les trois couches formant l'infrastructure de base décrite précédemment.

La figure 1.2 contient 3 types de couches :

– **Infrastructure de base (Discovery, Discription, Exchange) :** Ce sont les fondements techniques établis par l'architecture de référence. Nous distinguons les

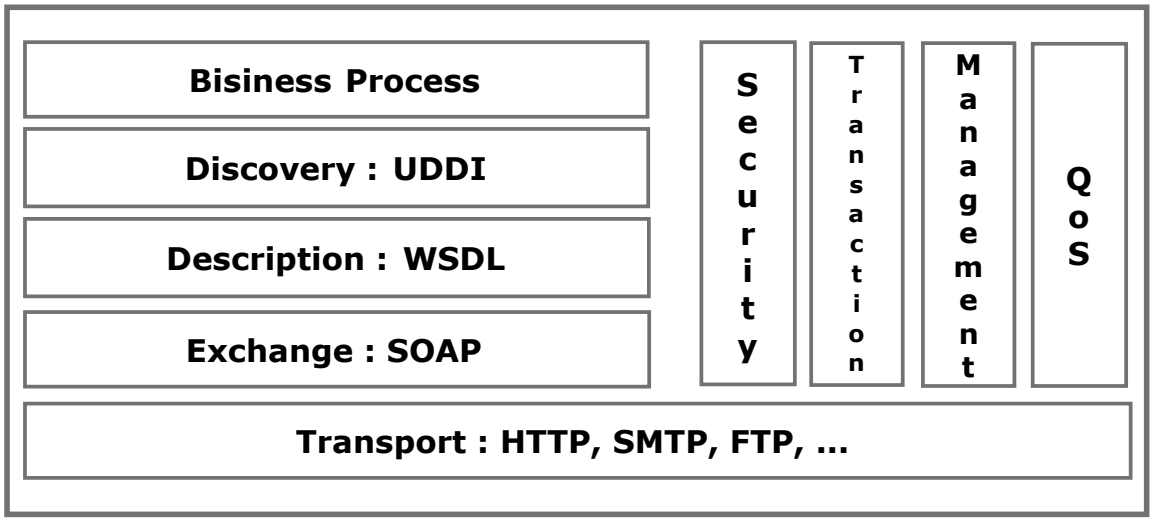

FIGURE 1.2 – Architecture étendue

échanges des messages établis par SOAP, la description de service par WSDL et la recherche de services Web que les organisations souhaitent utiliser via le registre UDDI ;

- **Couches transversales (Security, Transactions, management, QoS [10]) :** Ce sont ces couches qui rendent viable l'utilisation effective des services Web dans le monde industriel ;
- **Couche Business Processus (BusinessProcess) :** Cette couche supérieure permet l'intégration de services Web, elle établit la représentation d'un « BusinessProcess » comme un ensemble de service Web. De plus, la description de l'utilisation des différents services composant ce service est disponible par l'intermédiaire de cette couche.

1.4 Les standards de base des services Web

1.4.1 Le protocole SOAP (Simple Object Access Protocol)

SOAP est un protocole de transmission de messages basé sur XML. C'est un standard recommandé par le World Wide Web Consortium qui permet aux services Web d'échanger des informations dans un environnement décentralisé et distribué tout en s'affranchissant des plates-formes et des langages de programmation utilisés. Il définit un ensemble de règles pour structurer des messages qui peuvent être utilisés dans de simples transmissions unidirectionnelles, mais il est particulièrement utile pour exécuter des dialogues requête-réponse RPC. Il est assez léger, simple et facile à déployer, extensible et ouvert, mais ne garde pas la trace des requêtes envoyées par différents clients vers les services Web invoqués [28].

10. Quality of Service

SOAP utilise les protocoles HTTP [11] et XML. HTTP comme mécanisme d'invocation de méthodes en utilisant une des balises spécifiques pour indiquer la présence de SOAP. Cela permet de franchir aisément les pare-feu et proxys et facilite le traitement en cas de filtrage. XML pour structurer les requêtes et les réponses, on indique les paramètres des méthodes, les valeurs de retours et les éventuelles erreurs de traitements.

XML joue un rôle prépondérant dans SOAP, on peut distinguer plusieurs éléments :

- Une enveloppe, expliquant comment la requête doit être traitée et présentant les éléments contenus dans le message.
- Un ensemble de règles de codage, permettant de différencier les types de données transmises.
- Une convention de représentation, permettant de représenter les appels aux procédures de traitement et les réponses.

1.4.1.1 Les messages SOAP :

Les messages SOAP sont des messages XML, ils se composent de trois éléments : Un élément racine appelé Enveloppe et deux autres éléments intégrés à l'enveloppe, appelés En-tête (optionnel) et Corps (obligatoire), la figure 1.3 illustre la structure d'un message SOAP.

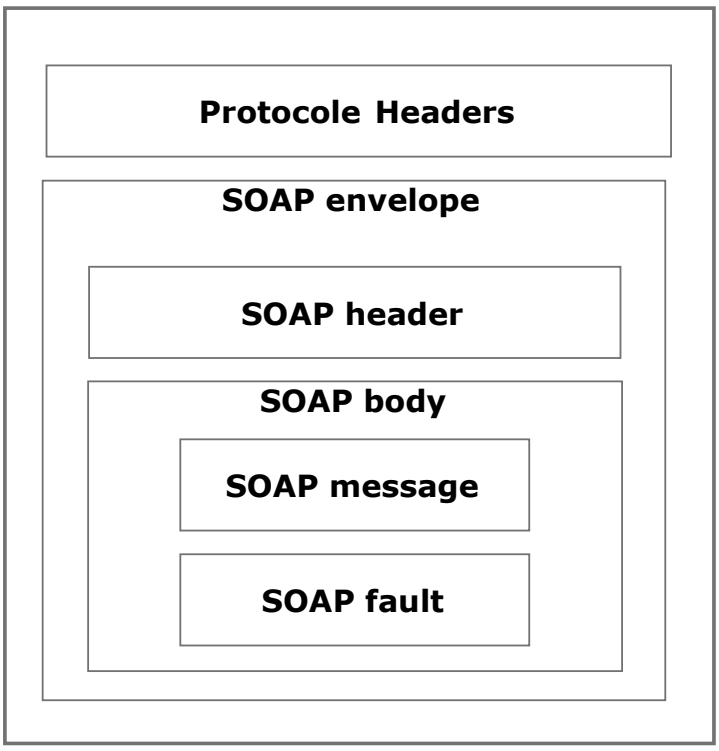

FIGURE 1.3 – Message SOAP

Les éléments de SOAP sont [72] :

11. HyperText Transfer Protocol : protocole de transfert hypertexte

- **Protocol headers :** Définit le protocole de transport exemple HTTP.
- **Enveloppe du message SOAP :** L'enveloppe SOAP marquée par la balise <Envelope>. Elle est obligatoire et englobe les deux autres éléments d'un message SOAP (Header et Body), et définit le cadre pour décrire ce qui est dans le message et comment le traiter. Le protocole SOAP permet de spécifier la version de SOAP utilisée, en utilisant un espace de nom.
- **Entête du message SOAP :** L'entête est un élément facultatif imbriqué dans l'enveloppe, marqué par la balise <Header>, permet de passer dans le message SOAP des informations complémentaires, c'est-à-dire des caractéristiques et des fonctionnalités additionnelles sur ce même message. Les règles d'encodages sont placées dans l'entête et servent à exprimer et définir le mécanisme de représentation des données.
- **Corps du message SOAP :** Le corps du message SOAP est obligatoire, marquée par la balise <Body> et permet de transmettre les requêtes et les réponses entre les systèmes, il est composé d'un ou de plusieurs sous éléments, qui sont : FAULT et MESSAGE
  - **Elément FAULT :** Il permet d'indiquer les défaillances de transmission des massages SOAP. Il renvoie des informations sur le type d'erreur, une description de l'erreur et l'adresse du serveur SOAP qui a généré l'erreur ;
  - **Élément MESSAGE :** Il contient les données à transmettre via le protocole SOAP.

Les messages SOAP sont des transmissions fondamentalement à sens unique d'un expéditeur à un récepteur. Lorsqu'une transmission d'un message commence, un message SOAP est généré. Ce message est envoyé à partir d'une entité appelée le « SOAP Sender », localisé dans un « SOAP Nud ». Le message est soit transmis ou non transmis à plusieurs n œud(nuds) intermédiaires « SOAP Intermediates ». Le processus se termine lorsque le message arrive au « SOAP Receiver ». Le chemin suivi par un message SOAP est nommé « Message Path ».

1.4.2 WSDL (Web Services Description Language) :

WSDL est le langage de la famille XML permettant de décrire et de publier les interfaces et protocoles des services Web d'une manière standard. L'interface d'un service Web décrit le fonctionnement d'un service Web, et cache l'implémentation du service Web, donc elle est indispensable pour pouvoir invoquer un service Web par une application cliente, ou un autre service Web permettant une utilisation indépendante de la plateforme utilisée ainsi que du langage utilisé [72]. L'interface intègre les composants fondamentaux suivants :

- Informations sur toutes les fonctions disponibles publiquement ;
- Définitions abstraites des données à transmettre ;
- Informations sur le type de données pour tous les messages XML ;
- Informations obligatoires sur le protocole de transfert à utiliser spécifiquement ;
- Informations de type Adresse pour localiser le service à spécifier ;

Le langage WSDL présente un format commun pour la description et la publication des interfaces et protocoles relatifs aux services Web. Une description WSDL d'un service Web est faite sur deux niveaux, niveau abstrait et niveau concret. La figure 1.4 représente ces deux niveaux. Au niveau abstrait, la description du service Web consiste à définir les éléments de l'interface du service Web tel que : Types de données (DataTypes), les messages (Message), les opérations (Operation), les types de ports (PortType), et les liaisons (Bindings), ces parties décrivent des informations abstraites indépendantes au contexte de mise en œuvre. On y trouve : Les types de données envoyées et reçues ; les opérations utilisables et le protocole qui sera utilisé [30].

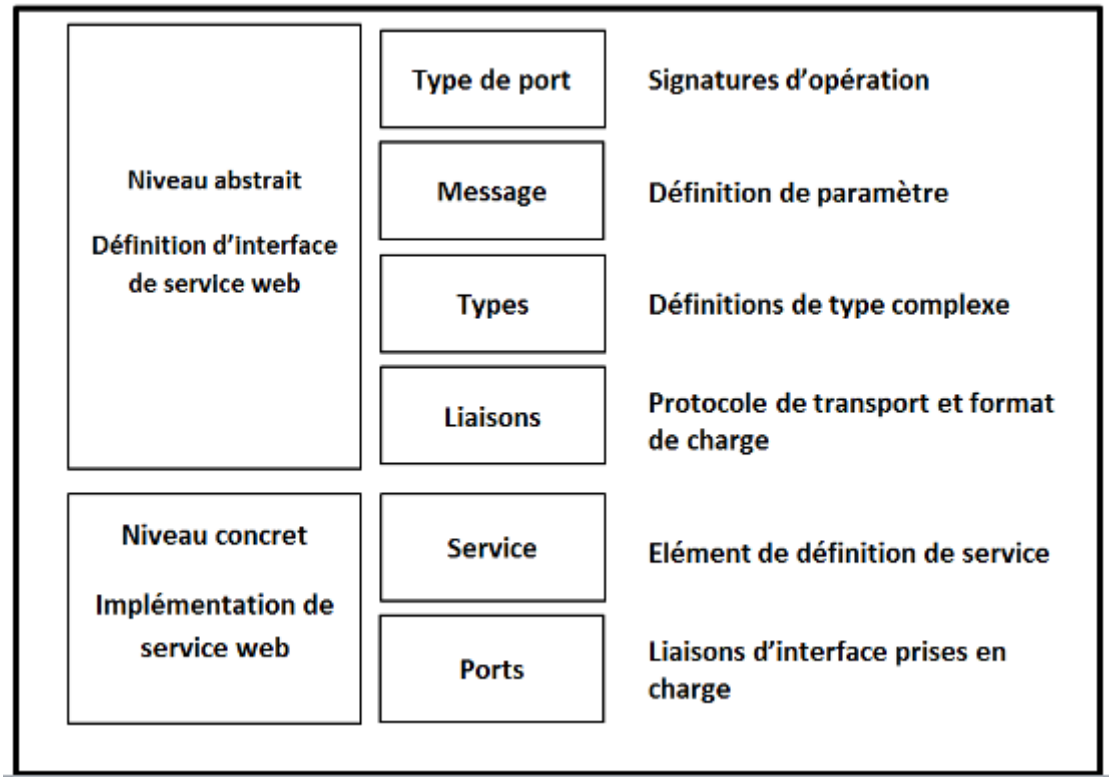

FIGURE 1.4 – Structure d'un fichier WSDL

- **Types de données :** « Data types » est l'élément qui définit les types de données utilisées dans les messages échangés par le service Web. Une fois définie, les « Data types », ou type peuvent être référencés dans tout type de message.
- **Messages :** L'élément « Message » spécifie les types d'opérations supportées par le service Web, il permet d'incorporer une séquence de messages corrélés sans avoir à spécifier les caractéristiques du flux de données, par exemple, un message Input

et un message Output corrélé sont mis en correspondance dans une seule opération de type « Request/Response ».

- **Opérations :** L'élément « Operation » spécifie les types d'opérations supportées par le service Web, il permet d'incorporer une séquence de messages corrélés sans avoir à spécifier les caractéristiques du flux de données.
- **PortType :** Le « PortType » est un groupement logique ou une collection d'opérations supportées par un ou plusieurs protocoles de transport, il est analogique à une définition d'un objet contenant un ensemble de méthodes.
- **Liaisons :** Décrit la façon dont un type de port est mis en œuvre pour un protocole particulier (HTTP par exemple), et un mode d'invocation (SOAP par exemple). Cette description est faite par un ensemble donné d'opérations abstraites, pour un type de port, on peut avoir plusieurs liaisons, pour différencier les modes d'invocation ou de transport des différentes opérations.

Au niveau concret, le service Web est défini grâce aux deux éléments ; port et Service. Ces deux derniers décrivent des informations liées à un usage contextuel du service Web. On y trouve : L'adresse du fournisseur implémentant le service, et le service qui est représenté par les adresses des fournisseurs.

- **Elément Port :** L'élément « Port », dans la partie concrète, spécifie une adresse URL qui correspond à l'implémentation du service Web par un fournisseur, et identifie un ou plusieurs « Bindings » (ou liaisons) aux protocoles de transports (HTTP, SMTP [12], FTP [13]. . .) Pour un « Port-Type » donné. La séparation du protocole de transport de la définition du « PortType » permet à un service Web d'être valable à travers plusieurs protocoles de transport, sans avoir à redéfinir l'ensemble du fichier WSDL.
- **Élément Service :** Spécifie l'adresse complète du service Web, et permet à un point d'accès d'une application distante de choisir à exposer de multiples catégories d'opérations pour divers types d'interactions.

1.4.2.1 Document WSDL

Un document WSDL est constitué de plusieurs définitions. Ces dernières définissent un service comme un groupe d'un ou de plusieurs ports. Chaque port est associé à une liaison spécifique. C'est cette liaison qui détermine la manière dont un ensemble abstrait d'opérations et de messages est lié à un port selon un protocole particulier. Une liaison établit une correspondance entre un protocole donné et un type de port. Un type de port se compose d'une ou de plusieurs opérations, qui représentent un ensemble abstrait d'opérations que peut réaliser le service. Chaque opération est constituée d'un ensemble

12. Simple Mail Transfer Protocol : http ://tools.ietf.org/html/rfc4409
13. File Transfer Protocol : http ://tools.ietf.org/html/rfc3659

de messages abstrait qui représente les données communiquées au cours de l'opération. Chaque message contient une ou plusieurs données définies par des types [50].

un document WSDL apparait comme une série de définitions. Il y a un élément définition à la racine qui peut en encapsuler d'autres. La grammaire d'un document WSDL se présente sous la forme du schéma XML.

1.4.3 UDDI (Universal Description, Discovery and Integration)

UDDI définit les mécanismes permettant de répertorier les services Web. Ce standard gère l'information relative à la publication, la découverte et l'utilisation d'un service Web. UDDI définit un registre des services Web sous un format XML. Les organisations publient les informations décrivant leurs services Web dans l'annuaire, et l'application client ayant besoin d'un certain service, consultent cet annuaire pour la recherche des informations concernant le service Web qui fournit le service désiré, pour une éventuelle interaction ; ainsi, UDDI a été créé pour faciliter la découverte de services Web. L'annuaire UDDI repose sur le protocole SOAP, les requêtes et les réponses sont des messages SOAP.

1.4.3.1 Principe d'UDDI

UDDI permet de classer et de rechercher des services Web. Si l'accès à un service Web est trop élevé, le recours aux services Web ne devient plus intéressant. C'est pourquoi le principe d'annuaire universel a été mis en place.
Cet annuaire est composé de deux bibliothèques :

- **API de requête :** Cela consiste à effectuer des interrogations, des recherches d'informations sur les entrées d'un UDDI. Tout utilisateur a la possibilité d'effectuer des recherches et donc de mettre en œuvre cette API.
- **API de publication :** Pour permettre la publication ou la suppression d'un service dans un annuaire. Cette API impose des autorisations d'accès et est destinée aux fournisseurs de services et aux entreprises désirant publier des informations les concernant.

1.4.3.2 Composants de l'annuaire UDDI

L'UDDI est subdivisé en deux parties principales : Partie publication ou inscription, et partie découverte. La partie publication regroupe l'ensemble des informations relatives aux entreprises et à leurs services. Ces informations sont introduites via une API d'enregistrement. La partie découverte facilite la recherche d'information contenue dans UDDI grâce à l'API SOAP. L'UDDI peut être vu comme un annuaire contenant [62] :

- **Les pages blanches :** Noms, adresses, identifiants et contacts des entreprises enregistrées. Ces informations sont décrites dans des entités de type « BusinessEntity

». Cette description inclut des informations de catégorisation permettant de faire des recherches spécifiques dépendant du métier de l'entreprise.

- **Les pages jaunes :** Donnent les détails sur le métier des entreprises et les services qu'elles proposent. Ces informations sont décrites dans des entités de type « BusinessService ».
- **Les pages vertes :** Contiennent des informations techniques du service offert, la manière d'interagir avec le service, une définition du « BusinessProcess », un pointeur vers le fichier WSDL et une clé unique identifiant le service.

la structure de données du registre UDDI contient cinq types de données :

- **BusinessEntity :** Les informations concernant l'entreprise qui publie ses services Web dans l'annuaire et le type de services qu'elle offre sont contenues dans la structure « BusinessEntity » et correspondent notamment aux pages blanches concernant l'entreprise.
- **BusinesService :** L'élément « BusinessService » contient des informations sur les services métiers. Il s'agit des informations de description des services Web référencés : Nom du service, sa description, son code. Une entreprise peut enregistrer plusieurs services. Le type d'information contenu dans l'élément « BusinessService » correspond aux informations pages jaunes d'une entreprise.
- **BindingTemplete (Informations de liaison) :** Les informations de liaison contiennent des informations techniques sur un service Web. Elles servent également de pages vertes de l'annuaire UDDI. Il s'agit des informations indiquant les références aux « tModels » désignant les spécifications de l'interface pour un service Web, ainsi le point de terminaison (adresse Internet) de ce dernier. Ces informations aident le client à se connecter puis à invoquer le service désiré. Un service peut avoir plus d'un modèle de liaison.
- **tModel (Informations techniques) :** La structure « modèle » a pour rôle de décrire les spécifications des services Web à enregistrer. Elle comporte les informations permettant de connaître les normes auxquelles le service est conforme afin d'avoir une description précise du service. Le nom doit présenter suivant le format URI et doit pointer vers l'emplacement du fichier correspondant, généralement au format WSDL.
- **PublishAssertion (Assertion contractuelle entre partenaires) :** Met en correspondance deux ou plusieurs structures « BusinessEntity » selon le type de relation (filiale de, département de). Cette structure comporte les assertions contractuelles entre organismes pour les services publiés. Ces assertions représentent un ensemble de règles contractuelles d'invocation de services représentées sous la forme de protocoles entre deux partenaires métiers. Chaque rôle entre partenaires est défini à travers ces assertions.

1.5 Service Web sémantique

Les informations du Web actuel, sont compréhensibles que par les humains, les machines peuvent lire ces informations, mais elles sont incapables de les interpréter. Le Web est limité, et ne permet pas un réel partage du savoir ; tout au plus, il permet de présenter des connaissances, mais en aucun cas de ne les rendre directement utilisables.

selon Tim Berners-Lee [12] : « *Le Web sémantique n'est pas un Web distinct, mais bien un prolongement du Web que l'on connaît, dans lequel, on attribue à l'information une signification clairement définie, ce qui permet aux ordinateurs et aux humains de travailler en plus étroite collaboration.* »

Le Web actuel est essentiellement syntaxique, dans le sens que la structure des documents (ou ressources au sens large) est bien définie, mais que son contenu reste quasi inaccessible aux traitements des machines. Seuls les humains peuvent interpréter leurs contenus. Le Web sémantique a pour but de rendre les ressources du Web actuel non seulement compréhensibles par les humains, mais aussi interprétables par des machines. Le Web sémantique est une infrastructure pour permettre l'utilisation de connaissances formalisées (ontologies) en plus du contenu informel actuel du Web.

1.5.1 Architecture du Web sémantique

L'évolution des travaux réalisés dans le cadre du Web sémantique est marquée par différents niveaux de complexité. Si, les plus élémentaires consistent en une simple affectation de métadonnées aux ressources utilisées, d'autres se basent sur des inférences très complexes et permettent l'exploitation de ressources hétérogènes [9]. Quel que soit le niveau de complexité, ces applications reposent toutes sur une architecture en couches commune exprimée par la figure 1.5, recommandée par le W3C. L'architecture repose sur des technologies de structuration de documents fondées sur XML.

le niveau Nommage et Adressage : Le Web repose sur un concept important qu'est l'URI. Tout ce qui est disponible sur Internet doit être identifié par un URI. L'URI est un protocole simple et extensible pour identifier, d'une manière unique et uniforme, toute ressource sur le Web. Il s'agit d'un aspect central de l'infrastructure, c'est pour cette raison que cet élément se trouve à la base de l'architecture. D'une façon générale, le terme identificateur désigne une clé capable de référencer un objet ayant une identité.
Dans le cas du Web sémantique, l'URI est une séquence de caractères avec une syntaxe restreinte, qui permet d'identifier toute ressource utilisée dans le cadre d'une application Web sémantique. On entend par ressource tout objet ayant une identité, tel qu'un document électronique, une page HTML, un fichier, une image, une vidéo... etc. Une ressource,

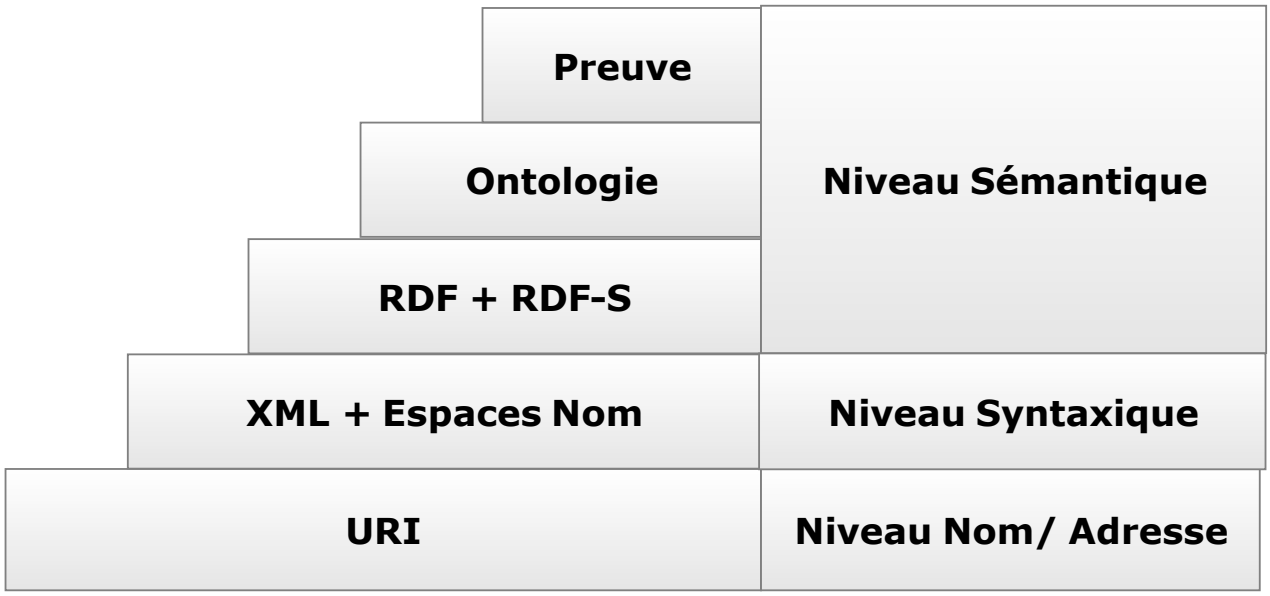

FIGURE 1.5 – Architecture du Web sémantique

n'est pas nécessairement un objet faisant partie du réseau, il peut s'agir d'un internaute, d'un livre, d'une corporation, ..., etc. [64]

le niveau syntaxique : Le niveau syntaxique est le niveau de la structuration des documents. La spécification de la structure logique des documents repose sur XML. Il fait le lien entre les documents et les logiciels qui les utilisent. En effet, le but de XML est de faciliter la diffusion et l'échange d'informations sur Internet.

le niveau sémantique : Il est nécessaire de rendre les données interopérables par une machine par l'association d'une sémantique formelle. Un langage est interprétable par une machine s'il dispose d'informations sémantiques. Cela signifie que les symboles et la structure du langage font référence à un modèle sémantique [46].
Le sens existe au travers des relations entre les concepts. Le W3C propose d'ajouter des métadonnées et/ou des annotations permettant de décrire en termes de contenu chacune des ressources disponibles. La différence entre métadonnée et annotation réside dans le fait qu'une métadonnée serait attachée à une ressource et est définie suivant un schéma, alors que l'annotation serait située au sein de la ressource est mise en place au cours d'un processus d'annotation [56].
RDF [14] est un standard permettant la mise en place de descriptions simples. XML est à la syntaxe, ce que RDF est à la sémantique. RDF Schéma permet ensuite de combiner ces descriptions en un seul vocabulaire. À tout ceci, il manque la possibilité de décrire des vocabulaires spécifiques à des domaines bien particuliers. C'est là que les ontologies écrites avec d'autres langages jouent leur rôle.

14. Resource Description Framework

1.5.2 Les ontologies

Dans le cadre de l'intelligence artificielle, Neeches et ses collègues [61] furent les premiers à proposer une définition à savoir : « *Une ontologie définit les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui indiquent comment combiner les termes et les relations de façon à pouvoir étendre le vocabulaire* ».

Grüber et son équipe à Stanford [43] proposent la définition suivante : « Une ontologie est une spécification formelle, explicite d'une conceptualisation partagée ». Il précise que cette définition est faite dans le contexte du partage et de la réutilisation des connaissances, et que ce qui importe est ce pour quoi (quel usage) une ontologie est faite.
Une explication a été apportée par [57] à la définition donnée par Grüber : "Une conceptualisation fait référence à un modèle abstrait de certains phénomènes du monde, modèle qui identifie les concepts pertinents de ce phénomène. Explicite signifie que le type des concepts utilisés et les contraintes sur leur utilisation sont explicitement définis. Formelle se réfère au fait que l'ontologie doit être compréhensible par les machines. Partagée reflète la notion de connaissance consensuelle décrite par l'ontologie, c'est-à-dire qu'elle n'est pas restreinte au point de vue de certains individus seulement, mais reflète un point de vue plus général, partagé et accepté par un groupe.

Donc, une ontologie est une structure de donnée opérationnelle qui rend compte des concepts d'un domaine et de leurs relations. Le développement des ontologies croissant en Intelligence Artificielle vient de leur intérêt pour associer du sens à des ressources textuelles, pour localiser et gérer des connaissances dans diverses applications [38].

1.5.2.1 Types d'ontologies

Les ontologies sont de nature diverses. Elles ont été classées selon leurs objets de conceptualisation par de nombreux auteurs [66, 80, 77]. Les méthodes en ingénierie des connaissances ont répertorié plusieurs types d'ontologie :

Ontologie générique (dite de haut niveau) : Repére et organise les concepts les plus abstraits du domaine, exemple ontologie de SOWA, Wordnet ;

Ontologie de domaine (la plus utilisée) : Spécifie le vocabulaire du domaine traité ; les concepts clés, l'ensemble des relations, les attributs, les instances relatifs au domaine.

Ontologie application : Décrit la structure des connaissances nécessaires à la réalisation d'une tâche particulière, elle permet aux experts d'utiliser le même langage que celui de l'application ;

Ontologie de représentation : Spécifie un formalisme de description qui fourni une structure de représentation des primitives pour décrire les concepts des ontologies de domaine et des ontologies génériques, exemple 'La farme Ontology [43]' ;

Ontologie de méthode de résolution de problèmes : Décrit le processus de raisonnement d'une façon indépendante d'un domaine ou d'une application donnée.

1.5.2.2 Caractéristique des ontologies

Les ontologies possèdent des caractéristiques fondamentales suivantes [5] :

- **Les ontologies sont formelles :** Ceci signifie qu'elles sont exprimées dans une langue qui a une syntaxe clairement définie, et une base mathématique pour leur signification. Comme, les concepts sont exprimés formellement, ils peuvent être traités par des programmes informatiques.
- **Les ontologies sont lisibles par les humains :** Ceci signifie qu'elles peuvent être développées, partagées, et comprises non seulement par des programmes informatiques, mais aussi par les communautés d'experts du domaine ainsi que des utilisateurs potentiels.
- **Les ontologies sont vastes :** Elles sont conçues dans le but d'inclure toute la signification appropriée des concepts liés à un domaine et simplement celle requise pour une application particulière. Cela veut dire que, si toute la signification des concepts est capturée par une ontologie, elle peut être comprise, modifiée, et contrôlée par n'importe quel expert de domaine.
- **Les ontologies sont partageables :** Elles sont construites sur la base de bibliothèques communes de concepts fondamentaux et sont utilisables à travers de multiples domaines d'application. Ceci facilite la combinaison des ontologies développées séparément pour permettre la communication entre les systèmes d'information, qui doivent partager des informations basées sur des concepts communs.

1.5.2.3 Composantes d'une ontologie

Les connaissances traduites par une ontologie sont définies par les éléments suivants [55] : 1) Concepts ; 2) Relations ; 3) Fonctions ; 4) Axiomes ; 5) Instances.

- Les concepts, aussi appelés termes ou classes de l'ontologie, correspondent aux abstractions pertinentes d'un segment de la réalité (le domaine du problème), retenue en fonction des objectifs qu'on se donne et de l'application envisagée pour l'ontologie. Selon [55] ces concepts peuvent être classifiés selon plusieurs dimensions : 1) niveau d'abstraction (concret ou abstrait) ; 2) atomicité (élémentaire ou composée) ; 3) niveau de réalité (réel ou fictif).

- Les relations traduisent les associations (pertinentes) existantes entre les concepts présents dans le segment analysé de la réalité. Ces relations incluent les associations suivantes : 1) Sous-classe-de (généralisation spécialisation) ; 2) Partie-de (agrégation ou composition) ; 3) Associée-à ; 4) Instance-de, etc. Ces relations nous permettent d'apercevoir la structuration et l'interrelation des concepts, les uns par rapport aux autres. Les fonctions constituent des cas particuliers de relations, dans laquelle un élément de la relation, le nième est défini en fonction des n-1 éléments précédents.
- Les axiomes constituent des assertions, acceptées comme vraies, à propos des abstractions du domaine traduites par l'ontologie.
- Les instances constituent la définition extensionnelle de l'ontologie ; elles sont utilisées pour représenter des éléments dans un domaine.

1.5.2.4 Rôle des ontologies

Les ontologies servent à la représentation des données échangées dans un domaine particulier, afin de faciliter la communication interne du système informatique et externe entre les différents acteurs du domaine. Leur utilisation peut varier de la représentation des données à la recherche d'informations. Les ontologies ont été employées dans divers domaines et pour différents objectifs. Elles ont également porté leurs fruits au sein des systèmes à bases de connaissances et du Web sémantique. Nous énumérons dans ce qui suit, les principaux rôles que peuvent jouer les ontologies [45] :

- **Communication :** Les ontologies permettent le partage de la compréhension et de la communication dans des contextes particuliers selon les besoins. Ainsi, on peut utiliser l'ontologie pour créer un réseau de relations qui définit les connexions entre les composants du système. Cette caractéristique de communication est offerte grâce à la non-ambiguïté des termes utilisés.
- **Ingénierie des systèmes :** L'ontologie peut servir divers aspects du développement des systèmes d'informations. Dans l'ingénierie des systèmes, elle joue un rôle important sur trois aspects : La spécification, la fiabilité et la réutilisation.
- **Interopérabilité :** Le développement et l'implantation d'une représentation explicite d'une compréhension partagée dans un domaine donné, peut améliorer la communication, qui à son tour permet une plus grande réutilisation, un partage plus large et une interopérabilité plus étendue [86]. L'interopérabilité est donc une spécialisation de la communication qui permet de répertorier les concepts que des applications peuvent s'échanger même si elles sont distantes et développées sur des bases différentes.
- **L'indexation et la recherche d'informations :** Dans le Web sémantique, les ontologies y sont utilisées pour déterminer les index conceptuels décrivant les ressources sur le Web.

1.5.3 Langages pour les services Web sémantique

Pour pouvoir exprimer de la sémantique avec les informations du Web, beaucoup de langages ont été élaborés tel que WSDL-S [15] [25], IRS-II [16] [40, 54], WSMF [17] [35, 39], WSMO [18] [58]...etc.

1.5.3.1 OWL-S (Ontology Web Language - Service)

OWL-S est une ontologie et un langage pour les services Web développé dans le cadre du projet DAML. OWL-S se base sur OWL qui permet de décrire des ontologies. Ainsi, OWL-S est une ontologie OWL particulière. OWL-S succède aux travaux antérieurs de DAML-S qui étaient basés sur DAML+OIL. OWL-S décrit un service à l'aide des trois classes suivantes (voir la Figure 1.6) [26, 27] :

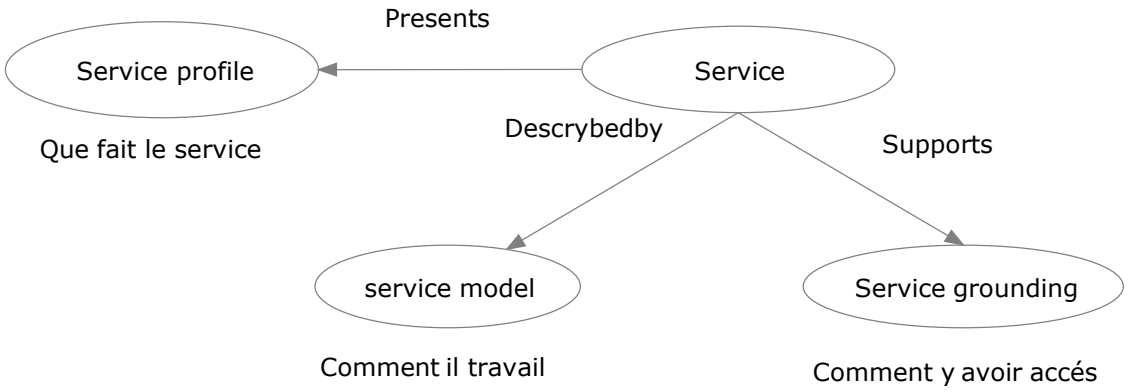

FIGURE 1.6 – Les classes de Owl-s

– **Service Profile :** Chaque instance de « Service » produit zéro ou plusieurs profils de services. Un service Profile exprime « Que fait un Service », aux fins des avertissements et sert comme un Template pour les requêtes de services, permettant ainsi la découverte et leurs arrangements. OWL-S fournit cette classe pour décrire un service Web. Cette classe « ServiceProfile » spécifie trois informations :
  - **Fournisseur du service :** Cette information précise les données nécessaires pour identifier et contacter le fournisseur du service Web.
  - **Description fonctionnelle :** Elle spécifie ce que l'on peut attendre du service en termes d'entrées attendues et de résultats produits en sortie. Les transformations d'informations sont représentées par des « Inputs » et des « Outputs ». Les « Inputs » et les « Outputs » font références à des classes d'OWL décrivant les types des instances à envoyer au service et aux réponses respectives attendues.

15. Web Language for Services semantic
16. Internet Reasoning Service
17. Web Service Modeling Framework
18. Web Service Modeling Ontology

- **Propriétés additionnelles :** Plusieurs propriétés sont utilisées pour qualifier le service Web. La première est la catégorie du service Web. La seconde est la qualité de service Web « QoS ».

– **Service Model :** Définis le fonctionnement du service Web. Les services Web peuvent être modélisés avec OWL-S en tant que processus. La classe ainsi définie est « Process », qui est une sous classe de « Service Model ». Pour décrire un processus, on spécifie ses entrées et sorties. Les transitions d'un état à un autre sont décrites par les préconditions et les effets de chaque processus. Un modèle peut être décrit par le lien « describedBy », un modèle de service qui expose comment un service fonctionne. Le modèle de service voit les interactions du service comme des processus. Un processus n'est pas nécessairement un programme à exécuter, mais plutôt des spécifications, qui permettent à un client d'agir avec un service.

– **ServiceGrounding :** Définit les détails techniques permettant d'accéder au service Web publié, tels les protocoles, les URIs, les messages envoyés... etc. Pour cela, il fournit les détails pour connecter la spécification abstraite et la spécification concrète. Il est effectivement nécessaire d'avoir une combinaison entre le grounding et WSDL. Le fonctionnement du grounding est assuré si et seulement si, les deux entités (WSDL et Grounding) sont présentées . Les concepts grounding de OWL-S sont compatibles aux concepts de « binding » du WSDL. Les deux classes « ServiceProfile » et « ServiceModel » d'une description OWL-S s'attachent à abstraire la représentation d'un service Web. Par contre, la classe « ServiceGrounding » est la forme concrète d'une représentation abstraite.

1.6 Composition des services Web :

Réellement, il n'est pas toujours facile de trouver des services Web qui repondent aux requêtes des utilisateurs. Puisque la plupart des services Web ont une fonctionnalité limitée. Pour remédier à ce problème, des services Web peuvent être regroupés pour ne former, du point de vue de l'utilisateur, qu'un seul service ; on parle alors de composition de services Web. La composition de services est une tâche complexe. Cette complexité est due principalement aux raisons suivantes [31] :

– Le nombre de services Web disponibles sur le Web est potentiellement très important et en constante augmentation ;

– Les services Web peuvent être créés et modifiés à la volée, le système de composition doit donc détecter et gérer ces changements ;

– Les services Web sont créés par des organisations différentes qui n'ont pas forcément les mêmes modèles de concepts pour décrire ces services.

1.6.1 Définition

La composition des services Web est le processus de construction de nouveaux services Web à valeur ajoutée, à partir de deux ou plusieurs services Web déjà présents et publiés sur le Web. Un service Web est dit composé ou composite lorsque son exécution implique des interactions avec d'autres services Web, et des changements des messages entre eux afin de faire appel à leurs fonctionnalités. La composition de services Web spécifie quels services ont besoin d'être invoqués, dans quel ordre et comment gérer les conditions d'interaction [31].

L'objectif principal de la composition de service Web est la possibilité de créer de nouvelles fonctionnalités d'un nouveau service Web, en combinant des fonctionnalités offertes par d'autres services Web existants. Elle implique la capacité de sélectionner, de coordonnée, d'interagir, et de faire interopérer des services Web existants.

1.6.2 Classification des services Web

Les services Web sont classés selon leur modèle d'interaction comme suit [36] :

1.6.2.1 Modèle de service Web atomique

Un service Web atomique est décrit comme une boite noire, c'est-à-dire qu'il est spécifié en terme de ses entrées /sorties ainsi que des préconditions et effet sans prendre en considération le fonctionnement du service.

Dans ce modèle d'interaction la composition des services Web est séquentielle ; l'exécution du service Web composite se fait par l'exécution séquentielle de chaque service Web composant.

1.6.2.2 Modèle de service Web comportemental

Les services Web basés sur un modèle comportemental sont souvent connus sous le nom de boite grise, ils sont décrits par l'ordre d'exécution de leurs opérations. Le genre de composition dans cette classe sera concurrentiel, cela veut dire qu'il peut y avoir plusieurs services Web actifs qui s'exécutent simultanément.

Nous distinguons deux types de modèles comportementaux :

(i) les services basés sur l'exécution : Le comportement d'un service Web est décrit par un séquence linéaire d'opérations, chaque opération donne une seule exécution ; dans chaque séquence, chaque point d'exécution a un seul et unique successeur, et *(ii)* les services basés sur un arbre : Les services Web basés sur l'arbre sont identiques à une

structure de n œuds qui décrit le comportement d'une exécution possible, chaque point d'exécution peut avoir plusieurs successeurs [11, 10, 24].

1.7 Avantage des services Web

Les services Web sont très flexibles, indépendants des langages de programmation et des systèmes d'exploitation. Cela leurs permettent de profiter de différents environnements et langages de développement par une publication, localisation, description et une invocation via XML.

Les services Web permettent aux applications et aux logiciels écrits dans différents langages de programmation, et évoluant sur différents systèmes d'exploitation de communiquer et d'échanger des données entre eux facilement.

Les services Web sont accessible à travers les pare-feu [19] sans modifier les réglages, et cela à l'aide du langage XML et les protocoles Internet standard comme HTTP sur le port 80, qui est généralement ouvert. Les protocoles et les formats de données sont au format texte dans la mesure du possible, facilitant ainsi la compréhension du fonctionnement global des échanges.

Les services Web peuvent interagir de manière synchrone (en temps réel) ou de manière asynchrone (décalé).

Les services Web peuvent être appelés par plusieurs manières (SOAP, XML-RPC, REST...).

1.8 Conclusion

Les Services Web ont été conçus pour répondre à un besoin d'interopérabilité. Leur objectif étant de faciliter la communication entre environnements de nature hétérogène. Face à cette problématique, la prise en compte de la sécurité n'a pas été systématique.

Les Services Web, par nature ouverts et interopérables, constituent des points d'entrée privilégiés pour des attaques. Afin de pallier à ces vulnérabilités potentielles, un certain nombre de mécanismes doivent être intégré tel que le contrôle d'accès.

19. Logiciel et/ou un matériel, permettant de faire respecter la politique de sécurité du réseau

Chapitre 2

Contrôle d'accès

Introduction

Un système qui partage des données entre plusieurs utilisateurs doit satisfaire une des exigences majeures qui est la protection des données contre la divulgation d'information non autorisées (Confidentialité), contre la modification non autorisée (Intégrité) et contre des attentes à la disponibilité (Déni de service). Cette protection ne sera assurée que si chaque accès aux données est contrôlé et bien évidemment tous les accès non autorisés doivent être impérativement bloqués. Cela est appelé le contrôle d'accès. Le développement d'une politique de contrôle d'accès repose sur la définition de politique de contrôle d'accès qui détermine qui a le droit d'effectuer quelle action sur quelle donnée. Le modèle veille à ce que les données ne soient accessibles que par des utilisateurs ayant le droit d'y accéder.

2.1 Généralités sur le contrôle d'accès

2.1.1 Définitions

2.1.1.1 Contrôle d'accès

Le contrôle d'accès est un mécanisme par lequel un système autorise ou interdit le droit à des entités actives d'accéder à des entités passives, ou d'effectuer des opérations ou des actions.

2.1.1.2 Entités

Les entités du système peuvent être réparties en deux ensembles : L'ensemble S des sujets, également appelés entités actives, qui correspondent aux entités qui effectuent les actions dans le système, et l'ensemble O des objets, également appelés entités passives,

qui subissent les actions. Les sujets et les objets sont généralement considérés comme des entités atomiques.

Ces deux ensembles ne sont pas nécessairement disjoints : Par exemple, un processus peut à la fois être un sujet et ainsi effectuer des opérations, et un objet, dans le cas où un autre processus tente de l'arrêter.

2.1.1.3 Accès

A est l'ensemble des modes d'accès qui caractérisent les différents types d'accès effectués par les sujets sur les objets. Cet ensemble contient généralement read, write, etc. Les approches classiques consistent à représenter un accès par un triplet (s, o, x), signifiant que le sujet s accède à l'objet o selon le mode d'accès x. Néanmoins, d'autres approches existent (on peut par exemple regrouper les modes d'accès, ou encore considérer les accès conjoints de sujets sur des objets). Afin de pouvoir prendre en compte ces différentes situations, on se limite à noter A l'ensemble de tous les accès, sans tenir compte de leur représentation.

2.1.1.4 Paramètres de sécurité

Il est souvent nécessaire d'associer de l'information aux entités afin de pouvoir exprimer la politique de sécurité, et également de décrire précisément le système. Ces informations sont construites à partir de ce qu'on appelle les paramètres de sécurité. Les paramètres de sécurité sont désignés par ρ.

Exemple de paramètre de sécurité : Chaque utilisateur et chaque ressource peut appartenir à une ou plusieurs équipes de travail (comptabilité, ressources humaines,...etc.), et nous introduisons le paramètre de sécurité $\rho_e = \{t_1, t_2, ..., t_k\}$, où chaque t_i représente un nom d'équipe

2.1.2 Notions sur le contrôle d'accès

2.1.2.1 Un système ouvert

Un système ouvert est principalement caractérisé par le fait que n'importe quel sujet s est a priori connecté. De plus, les moyens de communication relient directement les entités entre elles, sans passer par une entité centrale fiable.

2.1.2.2 Un système fermé

Dans les systèmes fermés, un sujet doit à la fois s'identifier et s'authentifier pour pouvoir se connecter au système, par le biais d'une autorité, généralement centralisée.

Ainsi, contrairement aux systèmes ouverts, tout sujet ne peut pas se connecter au système. De plus, les moyens de communication transitent généralement par une entité centrale, et on considère ainsi que les moyens de communication sont fiables.

2.1.2.3 Contrôle d'accès centralisé

Dans un système de contrôle d'accès centralisé, les autorisations ainsi que les interdictions sont administrées par une seule autorité. Ces systèmes ont l'avantage d'être faciles à administrer, mais les risques de sécurité dans de tels systèmes sont plus élevés, car si la seule autorité est attaquée, c'est tout le système qui devient vulnérable.

2.1.2.4 Contrôle d'accès distribué

Dans un système de contrôle accès distribué, les autorisations ainsi que les interdictions sont administrées par plusieurs autorités. Les avantages du contrôle d'accès distribué sont :

- Meilleure tolérance aux fautes : Si une autorité fournissant le contrôle d'accès tombe en panne, seuls les sujets servis par elle sont affectés. Les autres peuvent continuer de fonctionner normalement ;
- Amélioration de la sécurité : Il est plus difficile à un intrus de prendre le contrôle du système entier puisque pour cela il doit attaquer toutes les autorités qui forme le système de contrôle d'accès ;
- Meilleur scalabilité : Dans le cas de forte demande, il peut s'adapter et maintenir ses fonctionnalités et ses performances.

2.1.3 Concepts du contrôle d'accès

2.1.3.1 Les requêtes

Une demande d'accès est une requête (Request) qui spécifie les éléments suivants :

- L'identité des demandeurs qui seront appelés les sujets (Subjects) ;
- La ressource à accéder (Resource) ;
- L'action à effectuer (Action).

Ces éléments peuvent posséder des propriétés. Un sujet peut être défini par un identificateur, une institution à laquelle il appartient, un rôle etc. ; une ressource peut être caractérisée par un identificateur, un contenu structuré et un type ; idem pour l'action qui peut être définie par un identificateur.

Une requête peut contenir, en plus des informations concernant le sujet, la ressource et l'action, des informations optionnelles concernant l'environnement. Ces informations sont les propriétés qui ne sont associées ni au sujet, ni à la ressource, ni même à l'action, par exemple des informations sur un horaire ou une date ou même le lieu.

Un système de contrôle d'accès tient compte de la requête et des informations qu'elle contient pour prendre une décision. Une décision est générée à partir d'un ensemble de règles (rule). Les règles sont regroupées en politiques (policy). Les politiques peuvent être regroupées en des ensembles de politiques (policySet).

2.1.3.2 Règles de contrôle d'accès

Une règle de contrôle d'accès définit un ensemble de conditions et une décision. La décision peut être soit positive, pour permettre l'accès à la ressource (permit), soit négative, pour en refuser l'accès (deny). La décision constitue la réponse d'une règle dans le cas où les conditions imposées sont satisfaites. Une demande d'accès spécifie le sujet, la ressource à accéder, l'action à exécuter et les propriétés de l'environnement.

2.1.3.3 Politiques de contrôle d'accès

Une politique est une entité qui regroupe plusieurs règles de contrôle d'accès. En effet, l'objectif des langages de contrôle d'accès est de pouvoir fédérer plusieurs politiques, relatives à plusieurs systèmes répartis sur un réseau de communication. Il est clair qu'il faut grouper l'ensemble des règles d'autorisation en des ensembles, voir même des sous-ensembles, et ce, pour optimiser leur exploitation. Donc, si on a des règles de contrôle d'accès qui concernent les fichiers de notes dans une université et d'autres qui concernent les états de compte, il sera plus judicieux de les séparer en deux ensembles distincts. Ainsi, quand une demande de lecture du fichier de notes parviendra au système de contrôle d'accès, seules les règles concernées seront utilisées.

Une fois une politique est appliquée à un contexte de requête, toutes les règles qui sont contenues dans la politique sont appliquées. Une sélection plus fine est alors obtenue. Les cibles des règles limitent le nombre de règles appliquées dans une politique.

On distingue généralement deux grandes classes de politiques de sécurité : Les politiques discrétionnaires et les politiques obligatoires (mandataire).

2.1.3.4 Réponse

La réponse d'un système de contrôle d'accès peut être soit positive (permise) soit négative (deny). Par contre, nous pouvons faire face à des situations exceptionnelles. Quand une erreur inattendue survient ou que le système n'arrive pas à répondre, la réponse est indéterminée (Indetermined). Par contre, si tout se déroule bien, mais qu'aucune règle (politique ou ensemble de politiques) n'est appliquée, car les cibles ou les conditions ne sont pas vérifiées, alors la réponse sera non appliquée (Not Applicable). Notons que la seule réponse qui permette l'accès à une ressource est permise.

Une politique de sécurité est un ensemble de règles qui précisent comment gérer, protéger ou distribuer les informations ou ressources d'un système. Dans le cas du contrôle d'accès, une politique de sécurité est la définition (formelle ou informelle) des accès autorisés. Cette définition va dépendre de la notion d'entités, de l'information de sécurité disponible ainsi que de la caractérisation des accès.

2.2 Les modèles de contrôle d'accès aux systèmes d'information

Un modèle de contrôle d'accès peut être défini comme un formalisme (souvent mathématique) qui permet de développer et spécifier le comportement d'un système de manière exacte et de mieux le comprendre. Il permet aussi d'abstraire, donc de faciliter la compréhension d'une politique de sécurité et d'implémenter des mécanismes pour assurer certains objectifs de sécurité. Des exemples de ces objectifs sont la confidentialité dans le cas d'une compagnie privée où la sécurité peut être reliée à la divulgation des informations confidentielles (ou l'intégrité dans le cas où la sécurité peut être reliée à la modification des informations).

2.2.1 Modèle de contrôle d'accès discrétionnaire DAC (Discretionary Access Control)

TCSEC [1] définit le contrôle d'accès discrétionnaire comme : *"un moyen de restriction d'accès aux objets basé sur l'identité des sujets et/ou groupes auquel ils appartiennent. Les contrôles sont discrétionnaires dans le sens où le sujet est capable de transférer les permissions d'accès à d'autres sujets"*

Le modèle de contrôle d'accès discrétionnaire représente les politiques de contrôle d'accès sous forme d'un triplé $< sujet, objet, action >$ qui exprime que le sujet peut effectuer une certaine opération identifiée par l'action sur l'objet spécifié. Le triplet est appelé une règle d'autorisation.

2.2.1.1 Modèle de Lampson

La notion des droits d'accès spécifié par une matrice de contrôle d'accès a été introduite par Lampson dès 1971 [52]. Ce modèle peut être représenté par un triplet (S, O, M_{so}) où S désigne les sujets, O les objets et M_so la matrice de contrôle d'accès qui associe à chaque couple $(sujet\ s, objet\ o)$ un ensemble de droits d'accès. La matrice représentée par la

1. Trusted Computer System Evaluation Criteria

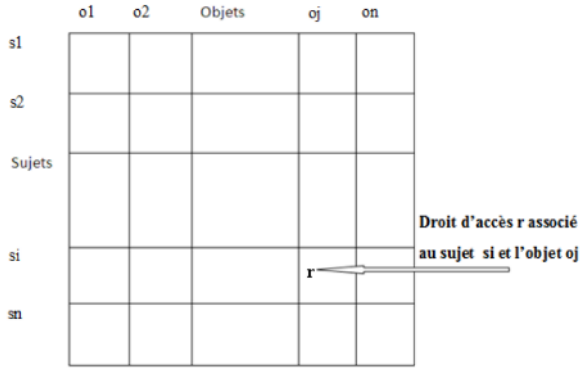

FIGURE 2.1 – Artefacts des méthodes formelles

figure 2.1 montre que le droit d'accès r est associé au sujet s_i et l'objet o_j. Cette matrice de contrôle d'accès ne permet pas de spécifier directement des interdictions. Ces dernières peuvent être parfois d'une grande utilité pour exprimer des exceptions.

2.2.1.2 Modèle de Harrison-Ruzzo-Ullman

Le modèle de Harrison Ruzzo Ullman [44] représente une amélioration du modèle de Lampson. Ce modèle de contrôle d'accès fournit des commandes pour attribuer les droits d'accès (read, write, own, ... etc.), ainsi que pour créer et supprimer les sujets et les objets. Ce modèle utilise une matrice M_{so} avec $s \in S$ et $o \in O$ et $M_{so} \in A$tel que A représente l'ensemble des actions.

Avantage du modèle DAC C'est un modèle flexible et facile a utiliser ;

Inconvénient du modèle DAC Les modèles discrétionnaires posent quelques problèmes comme :

- Vulnérabilité aux chevaux de Troie [8]. En effet, le problème majeur de la transitivité de la lecture qui en découle peut être une source de diffusion de chevaux de Troie.
- Ils n'assurent pas des propriétés de sécurité telles que la confidentialité et l'intégrité, car les permissions d'accès se font "à la discrétion" du propriétaire d'un objet. En effet, le propriétaire d'un objet peut attribuer à un sujet malveillant l'accès à des informations importantes parce qu'il n'a pas une vision globale du système.

2.2.2 Les modèles de contrôle d'accès obligatoire MAC(Mondatory Access Control)

Le contrôle d'accès discrétionnaire est généralement défini par opposition au contrôle d'accès obligatoire (MAC) [53] qui impose des règles incontournables garantissant l'atteinte des objectifs de sécurité visés. Dans ce type de contrôle d'accès, les sujets ne peuvent

pas intervenir dans l'attribution des droits d'accès. Ce contrôle d'accès est plus rigide que le contrôle d'accès discrétionnaire, mais plus sûr.

Définition Contrôle d'accès mandataire [69] *Le contrôle d'accès mandataire est exprimé en termes de niveaux de sécurité associée aux sujets et aux objets et à partir desquels sont dérivées les actions autorisées.*

Sécurité multiniveau Dans les systèmes multiutilisateurs, certaines informations sont susceptibles d'être plus importantes que d'autres. Une information sensible est une information qui, si elle est révélée à des personnes mal intentionnées, peut entraîner la perte d'un avantage. Il est donc nécessaire de mettre au point des moyens de sorte que seuls certains sujets ont accès à ces informations. Ainsi, les modèles de sécurité multiniveaux [21] permettent d'éviter la divulgation non autorisée de certaines informations aux utilisateurs "non fiables". Le mécanisme de mise en œuvre de cette politique est d'attribuer des niveaux de sécurité aux objets et aux sujets.

Dans ce modèle, un niveau de sécurité est attribué à chaque objet et sujet. Le niveau de sécurité d'un sujet est appelé Habilitation alors que le niveau de sécurité d'un objet est appelé Classification. Des exemples typiques de ces niveaux sont : "Unclassified", "Confidential", "Secret" ou "Top Secret".

2.2.2.1 Modèle de Bell-LaPadula

Le modèle de Bell-LaPadula [7], met en œuvre une politique mandataire multiniveaux, développée pour le DoD [2] des Etats-Unis. Le modèle de Bell-LaPadula a pour objectif de contrôler la confidentialité des données et d'éviter la propagation d'information d'un domaine de sécurité supérieur vers un domaine inférieur.

Le modèle se base sur la matrice de contrôle d'accès pour représenter les permissions d'accès du système, et définit en plus des sujets S et des objets O, un treillis de sécurité, noté $(SC, \leq)$ où SC est un ensemble de classes de sécurité et $\leq$ une relation de dominance. Chaque objet a ainsi un niveau de sécurité qui représente le danger que peut constituer la divulgation de l'information contenue dans cet objet, et chaque sujet a une habilitation qui désigne la confiance qui lui est accordée. La notion de dominance entre deux classes sc et sc' est définie pour que sc domine sc', noté $sc' \neq SC$, si le flot d'information de sc' vers sc est autorisé.

2. Department of Defense. http ://www.defense.gov/

2.2.2.2 Le modèle de Biba

Le modèle de Bell-LaPadula ne répond qu'à des besoins de confidentialité. C'est pour cette raison que d'autres politiques obligatoires ont été développées pour le maintien de l'intégrité. C'est le cas de la politique proposée par [20]. Celle-ci applique à l'intégrité un modèle ressemblant à celui de Bell-LaPadula pour la confidentialité. A chaque sujet s, on affecte un niveau d'intégrité $is(s)$, correspondant à la confiance qu'on a dans ce sujet et chaque objet o possède un niveau d'intégrité $io(o)$, correspondant au niveau d'intégrité du sujet qui la créé.

2.2.2.3 Le modèle de la muraille de chine

Le modèle de la muraille de chine [23] se compose de trois ensembles :
- C, l'ensemble des compagnies ;
- S, l'ensemble des sujets (analystes financiers) ;
- O, l'ensemble des objets (fichiers)

Le système classe l'information en trois niveaux hiérarchiques :
- Le niveau le plus bas contient les données de toutes les compagnies ;
- Le niveau intermédiaire, regroupe les objets qui concernent la même compagnie ;
- Le niveau supérieur regroupe les données des compagnies en compétition.

A chaque objet, associés le nom de la compagnie à laquelle il appartient ainsi que la classe de conflit d'intérêts qui contient ses données. Ce modèle considère deux fonctions :
- La fonction X, tel que $X(oi)$ désigne la classe de conflit d'intérêt de oi. Autrement dit, pour un objet donné, X fournit l'ensemble de toutes les compagnies en compétition autour de cet objet ;
- La fonction Y, tel que $Y(o_j)$ désigne l'ensemble des données de la compagnie de o_j.

2.2.2.4 Avantages des modèles MAC

- Contrairement aux modèles DAC, il n'y a pas de fuite d'informations ;
- Il est efficace dans un environnement centralisé tel que les systèmes d'exploitation.

2.2.2.5 Inconvénients des modèles MAC

L'inconvénient majeur du contrôle d'accès obligatoire est sa rigidité, car il est difficile de classer objets et sujets dans un nombre prédéfini de niveaux de sécurité. L'administration des modèles MAC est strictement centralisée, il n'est pas adapté aux systèmes réellement répartis.

2.2.3 Les modèles RBAC(Role Based Access Control)

De façon générale, un système de politique de contrôle mandataire peut être utilisé aisément dans une administration centrale. Cependant, il est statique et inapproprié pour sécuriser les interactions de diverses organisations indépendantes (Système distribué).
Pour combler ces lacunes, les politiques de contrôle d'accès à base de rôles (RBAC) [70] ont été introduites.
Le contrôle d'accès basé sur les rôles peut être considéré comme une approche alternative au contrôle d'accès obligatoire (MAC) et le contrôle d'accès discrétionnaire (DAC). Dans ce modèle, des permissions qui peuvent être représentées comme un couple (o, a) avec $o \in O$ (ensemble d'objets) et $a \in A$ (ensemble d'actions)), sont affectées à des rôles spécifiques au lieu d'être affectés directement à des sujets comme c'est le cas des modèles précédents. Ensuite, les sujets peuvent être attribués aux rôles qui découlent généralement de la structure d'une organisation. Ce modèle simplifie les opérations telles que l'ajout ou la suppression d'un sujet.
La figure 2.2 illustre le modèle RBAC :

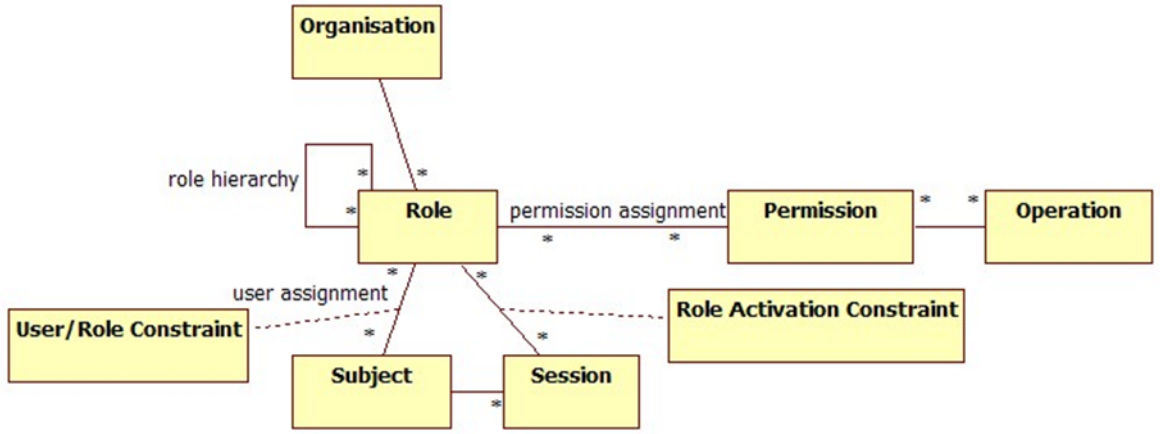

FIGURE 2.2 – Modèle RBAC [70]

2.2.3.1 Les modèles de références de RBAC

Des variantes du modèle RBAC ont été proposées, mais en réalité il existe quatre types de familles RBAC. $RBAC_0$ représente le modèle de base qui contient les éléments minimaux d'un modèle de contrôle d'accès (utilisateur, permission, session, rôle). Des extensions à $RBAC_0$ ont été faites par l'ajout de concept de la hiérarchie des rôles (un rôle peut hériter d'un autre rôle), cette extension a donné naissance à un modèle RBAC nommé $RBAC_1$. $RBAC_2$ ajoute un ensemble de contraintes tel que les contraintes de cardinalité, exclusion mutuelle, de temps ou de lieu. Le dernier modèle $RBAC_3$ combine entre les deux, soit un modèle hiérarchique avec contraintes. Les relations entre ces variantes sont

représentées par la figure 2.3.

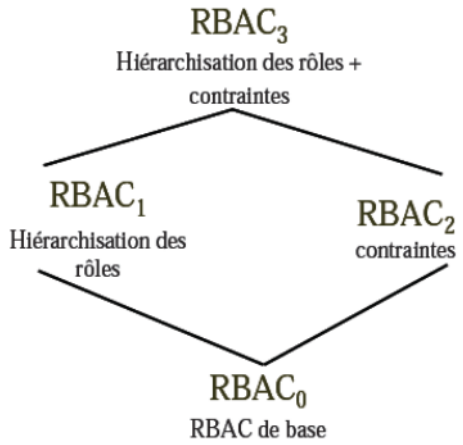

FIGURE 2.3 – Variante de RBAC

2.2.3.2 Modèle de base de RBAC

Les entités du modèle de base $RBAC_0$ sont : Les utilisateurs (U), les rôles (R), les permissions (P) ainsi que les sessions (S).

L'utilisateur : Dans ce modèle, l'utilisateur représente soit un humain, un agent autonome, un ordinateur,... etc.

Rôle : Un rôle est une fonction ou un titre de fonction au sein de l'organisation avec une certaine sémantique associée concernant l'autorité et la responsabilité attribuée à un membre du rôle.

Permission : Une permission est une autorisation d'un mode particulier d'accès à un ou plusieurs objets dans le système. Les termes autorisation, droit d'accès et privilège sont aussi utilisés dans la littérature pour désigner une permission.

Session : Chaque session est affectée à un seul utilisateur pour un ou plusieurs rôles, c'est à dire, un utilisateur établit une session au cours de laquelle l'utilisateur active un sous-ensemble de rôles dans lequel il est membre.

Inconvénient de RBAC : L'inconvénient de RBAC réside dans la difficulté de gérer des règles dépendantes du contexte, par exemple : "seuls les médecins traitants peuvent accéder aux informations médicales du dossier d'un patient" ou "les étudiants ont le droit d'accéder uniquement à leurs données personnelles". Une des solutions envisagées est de créer, par exemple, pour chaque étudiant un rôle privé. Théoriquement c'est une solution, mais en pratique cela n'est pas faisable, car il existe un très grand nombre d'étudiants et cela fait perdre à RBAC sa simplicité d'administration.

2.2.4 Les variantes de RBAC

2.2.4.1 Les modèle TMAC (Team-based Access Control)et CTMAC (Context TMAC)

L'objectif principale du modèle TMAC [85] est de fournir un contrôle d'accès dans le cas d'un travail collaboratif tout en exploitant la flexibilité du modèle de contrôle d'accès à base de rôle (RBAC). L'entité de base de TMAC, équipe ou "team", qui est une abstraction qui encapsule un ensemble d'utilisateurs, qui ont des rôles différents et qui collaborent dans le but d'accomplir une tâche commune ou d'atteindre un objectif commun. Les utilisateurs affectés à l'équipe devront bénéficier d'un accès à toutes les ressources de l'équipe. Cependant, les permissions exactes de chaque utilisateur sont déterminées par le rôle affecté et l'activité courante de l'équipe où il appartient.
Le concept TMAC distingue l'attribution et l'activation des permissions. Les permissions attribuées sont, par exemple, celles associées à un utilisateur lorsqu'il choisit un rôle au moment de la connexion ; les permissions activées sont celles qui dépendent des variables environnementales (lieu, temps, etc.), elles peuvent donc changer selon le contexte. Le pouvoir d'intégrer les informations contextuelles, lors de la décision d'accès, permet au modèle TMAC d'être flexible et d'exprimer des politiques d'accès pouvant fournir des permissions plus fines que celles de RBAC.

Selon TMAC, le contexte de collaboration d'une équipe donnée doit tenir compte de deux types de contexte :

- **Contexte utilisateur :** Les utilisateurs qui forment l'équipe à un moment donné ;
- **Contexte objet :** Les instances d'objets que l'équipe utilise pour accomplir sa tâche.

La figure 2.4 donne une description graphique de TMAC :

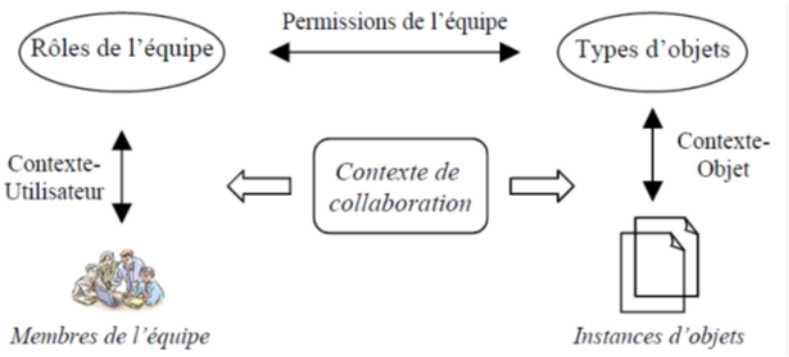

FIGURE 2.4 – TMAC [85]

C-TMAC est une amélioration de TMAC qui fournit, de manière explicite, les règles d'activation des permissions selon le contexte. C-TMAC utilise un hybride de RBAC et

TMAC, et consiste en cinq entités : Utilisateurs, rôles, permissions, équipes et contextes.

Durant une session, les permissions d'un utilisateur représentent l'union des permissions de tous les rôles qu'il a activé. Par ailleurs, dans le contexte sont incluses des informations concernant l'activité de son équipe, ainsi que des informations contextuelles comme le temps.

La figure 2.5 qui suit montre la structure de C-TMAC

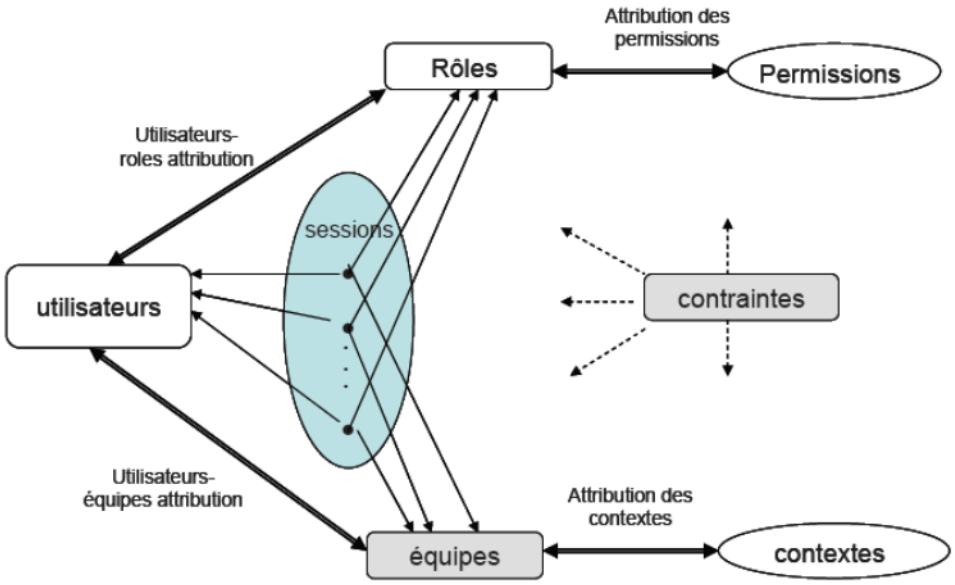

FIGURE 2.5 – CTMAC

Inconvénient Bien que les modèles TMAC et C-TMAC offrent un moyen de contrôle d'accès dans un cadre de travail collaboratif tout en bénéficiant de la flexibilité de RBAC, ces modèles souffrent d'un inconvénient majeur concernant la gestion des droits d'accès. Cela est lié, plus précisément, à l'ensemble des permissions offertes à l'équipe qui peut violer le principe du moindre privilège (chaque utilisateur dans un système se voit attribué l'ensemble des privilèges les plus restrictifs pour la réalisation des tâches autorisées). D'après ces modèles, un utilisateur rejoignant une équipe renforce les permissions de cette équipe en ajoutant les siens. Néanmoins, par exemple : Dans le secteur médical, bien que les professionnels de santé appartiennent à la même équipe dans le même hôpital, ils n'ont pas forcément les mêmes droits sur les parties du dossier médical du patient. Logiquement, il est évident que les permissions finales du médecin doivent être différentes de celle de l'infirmière même s'ils appartiennent à la même équipe.

2.2.4.2 Le modèles TRBAC (temporal role-based access control model)

Dans les systèmes opérationnels, il est très fréquent que les droits d'accès des utilisateurs varient selon certaines contraintes de temps. Par exemple, un utilisateur n'a pas le droit d'effectuer certaines tâches en dehors de ses heures de travail. Le modèle TRBAC [13] est le modèle de base RBAC auquel on ajoute la contrainte de temps. Ces contraintes temporelles s'expriment en termes de temps auquel les actions doivent être prises en compte et en termes de période à l'intérieur de l'intervalle.
TRBAC introduit la notion de déclencheur (Trigerres), qui correspondent à des actions qui s'exécutent automatiquement en fonction du temps. Ces actions sont mises en disponibilité des rôles qui ont une contrainte de temps afin de les activer et les désactiver.

Le modèle TRBAC, cependant, ne peut pas gérer plusieurs autres contraintes temporelles importantes. Premièrement, le modèle n'inclut pas les contraintes temporelles pour les affectations utilisateur-rôle et rôle-autorisation. Il suppose que seul le rôle est activé et désactivé dans différents intervalles de temps.
Deuxièmement, le modèle TRBAC ne gère que les contraintes temporelles sur l'activation de rôle et ne comprend pas les contraintes sur les activations réelles des rôles par les utilisateurs. Ainsi, le modèle TRABC n'a pas séparé la notion de rôle accessible et du rôle activé. Par conséquent, le modèle TRBAC ne peut pas gérer de nombreuses contraintes qui sont liées à l'activation d'un rôle comme les contraintes sur la durée maximale de l'activation autorisée à un utilisateur, le nombre maximum d'activations d'un rôle par un seul utilisateur au sein d'un intervalle de temps donné, etc.
Troisièmement, le modèle TRBAC ne tient pas compte des contraintes de durée ainsi que les contraintes sur les activations effectives des rôles, il ne supporte pas la notion d'activation et de désactivation des contraintes. Les contraintes d'activation doivent être clairement définies par rapport au temps permis pour un rôle.

2.2.5 Le modèle GTRBAC (Generalized Temporal Role-Based Access Control Model)

La clé du modèle GTRBAC [49] est qu'il distingue entre les notions d'accessibilité d'un rôle (role enabling) et l'activation d'un rôle (role activation), une telle distinction donne la notion d'état d'un rôle, la figure 2.6 montre les états possibles d'un rôle dans le modèle GTRBAC. Un rôle peut être dans l'un des trois états suivants : enabled (accessible), disabled (inaccessible) et active (activé).
L'état non permis (disable) indique que le rôle ne peut pas être utilisé par aucune session d'utilisateur c.-à-d. que l'utilisateur ne peut pas acquérir les permissions associées à ce rôle. Un rôle dans l'état non permis (disabled) peut être permis (enable).
L'état permis indique que les utilisateurs qui sont autorisés à utiliser le rôle dans le temps

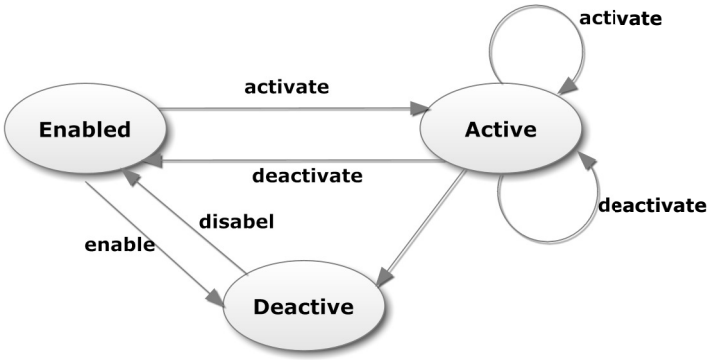

FIGURE 2.6 – Les états d'un rôle dans GTRBAC [49]

de la requête peuvent activer le rôle. Par la suite si l'utilisateur active le rôle, ce dernier devient un rôle dans l'état activé.
L'état activé d'un rôle implique qu'il y a au moins un utilisateur qui a activé le rôle. Une fois dans l'état activé, d'autres activations du même rôle ne changent pas son état.
Quand un rôle est dans l'état activé, après sa désactivation, l'état du rôle devient permis (enable) s'il y a qu'une seule session qui l'active, autrement il reste dans l'état actif. Un rôle dans l'état activé ou permis transite dans l'état non permis si un événement de désactivation se produit.

2.2.5.1 Modèle de contrôle d'accès Or-BAC (Organization Based Access Control)

Le modèle Or-BAC [34] vise à résoudre certains problèmes rencontrés par les modèles de contrôles d'accès existant et d'établir un modèle plus abstrait. Il s'intéresse non seulement aux permissions, mais aussi aux interdictions, obligations et recommandations. Or-BAC introduit au plus des concepts de rôles pour structurer les sujets, des concepts (notions) pour structurer les sujets et les actions. L'entité centrale de ce modèle est l'organisation.

L'organisation peut être un groupe de sujets jouant un rôle, ou une entité, hôpital, département ...etc. Le fait d'introduire la notion d'organisation dans le concept de base de ce modèle résout les problèmes des modèles RBAC et TBAC. Ces derniers définissent des relations binaires entre l'utilisateur et le rôle et l'utilisateur et l'équipe. Dans ce cas, un utilisateur qui a plusieurs rôles peut activer soit l'ensemble de ses rôles ou un sous-ensemble de ses rôles. Dans n'importe qu'elle équipe à laquelle il appartient. Dans la pratique, même si un utilisateur possède plusieurs rôles, il n'a pas forcément le droit de les activer dans toutes les équipes auxquelles il appartient

Les concepts de base de Or-Bac Les différents ensembles d'entités de Or-bac sont : Org (ensemble d'organisations), S (ensemble de sujets), A(ensemble d'actions),

V(ensemble de vues) et C(ensembles de contexte).
La figure 2.7 montre les différents éléments de Or-BAC où les rectangles représentent les entités et les ovales représentent les relations.

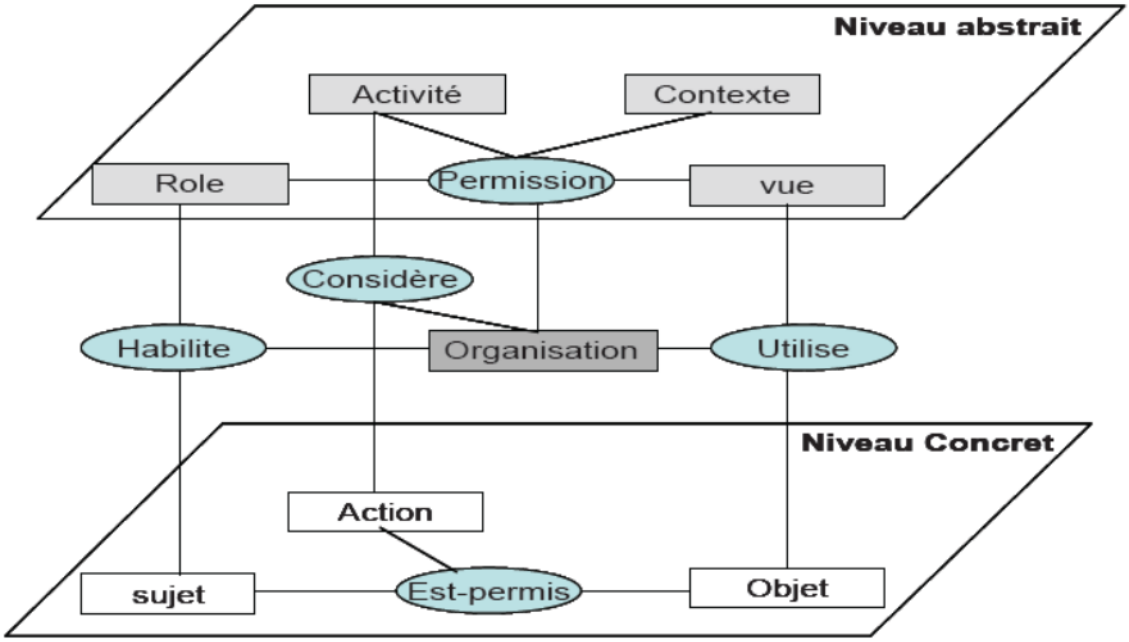

FIGURE 2.7 – Or BAC [34]

2.2.5.2 Inconvénient de Or-Bac

Dans un modèle Or-Bac il impossible de spécifier des règles relatives à plusieurs organisations indépendantes dans un système de collaboration en utilisant une politique Or-BAC unique. En outre, les autorisations ne peuvent pas être associées à des utilisateurs appartenant à d'autres organisations partenaires ou sous-organisations. Par conséquent, tandis que OrBAC peut exprimer la politique de sécurité d'une organisation, elle est insuffisante pour modéliser la collaboration et l'interopérabilité entre plusieurs organisations.

2.2.5.3 PolyOrBAC (poly Organization Based Access Control)

PolyOrbac [1] est une approche conçue pour modéliser et mettre en œuvre la sécurité et la collaboration entre organisations. PolyOrBAC se base sur deux composants principaux : Une extension du modèle de sécurité existante OrBAC pour prendre en compte les services Web, et une adaptation de la technologie des services Web pour permettre une collaboration sécurisée entre les organisations.

Le choix d'OrBAC pour modéliser les politiques de sécurité de chaque organisation est motivé par sa capacité d'abstraction qui facilite la spécification de politiques de sécurité

sans entrer dans le détail des utilisateurs et des ressources que doit contrôler l'organisation. La définition du concept d'organisation permet aussi de maîtriser la complexité en généralisant la notion d'héritage de rôle qui existait déjà dans RBAC. Également pour maîtriser la complexité et favoriser l'extensibilité (scalability), il est préférable de restreindre les interactions entre organisations à des relations bipartites, du type client-serveur, plutôt que de permettre des interactions sans contrainte, dont il serait plus difficile de spécifier et de mettre en œuvre. Le choix de la technologie des services Web comme modèle d'interaction dans PolyOrBAC résoud le problème de l'interaction.

Avec PolyOrBAC, il est donc nécessaire de définir des politiques de contrôles d'accès pour chacune des organisations participant dans le processus de collaboration. Dans une interaction par service Web, on peut distinguer deux organisations différentes : Le client qui utilise le service et le prestataire qui fournit ou met à disposition le service. Le client (en tant qu'organisation) doit contrôler les accès que ses propres utilisateurs peuvent vouloir faire à des services externes et le prestataire doit contrôler les accès à ses ressources locales par ses propres utilisateurs comme par l'organisation cliente.

Dans la pratique, deux grandes étapes dans PolyOrBac sont distinguées :

1. Publication et négociation des règles de collaboration et spécification des règles de contrôle d'accès qui leurs sont liées,
2. Accès en temps réel (à l'exécution) aux services distants.

2.2.6 Comparaison des modèles de contrôle d'accès étudiés

Le tableau 2.1 est une synthèse des modèles de contrôle d'accès pour les systèmes d'information, pour une meilleure lecture de ce tableau, on doit définir les concepts suivants : Un " + " dans une case signifie que le critère est supporté exclusivement, un " - " dans une case signifie que le critère peut être appliqué ou il n'a pas été détaillé. Un vide indique qu'aucun élément n'est donné.
La première ligne indique les modèles de contrôle d'accès, la première colonne indique les critères de comparaison.
Les critères proposés pour évaluer les différents modèles de contrôle d'accès sont regroupés en deux blocs :

1. La structuration des modèles :
   - Quels sont les concepts utilisés ou introduits pour structurer les droits. Les types de concepts utilisés dans la comparaison sont :
     - Rôle : Le modèle utilise le concept de rôle ;
     - Données : Les informations sont sous forme de données ;
     - Flux : Les informations sont sous forme de flux ;

- Niveau : Le modèle utilise les niveaux de sécurité ;
- Interdiction : Le modèle utilise en plus des autorisations, les interdictions ;

– Quels types de hiérarchies ont été proposés pour organiser les concepts. Les types de hiérarchie utilisés dans la comparaison sont :
- Limité : Il existe un nombre de niveaux hiérarchique limité ;
- Générale : Il existe plusieurs niveaux hiérarchiques, mais une seule hiérarchie ;
- Multiples : Le modèle utilise plusieurs hiérarchies.

– Quels sont les types de contexte utilisés ? Les différents types de concepts utilisés sont :
- Temps : Les autorisations ou les permissions dépendent du temps (l'horaire par exemple) ;
- Espace : Les autorisations ou les permissions dépendent du lieu par exemple ;
- Concept : Les autorisations ou les permissions dépendent du concept données ou flux ;
- Libre : Chaque autorisation ou permission dépende d'un contexte donnée ;

2. Les enrichissements proposés pour ces modèles :
   – Les catégories de contextes introduites dans l'expression des autorisations,

	DAC	**MAC**	**RBAC**	**TRBAC**	**GTRBAC**	**C-TMAC**	**OrBAC**
Concepts							
Rôles			+	+	+	+	+
Données	+						+
Flux						+	
Niveaux		+					
Interdictions			+		-		
Hiérarchie							
Limitées				+			
Générales			+	+		+	+
Multiples			-		+		-
Contexte							
Temps				+	+		-
Espace							-
Concept			+				
Libre							+

TABLE 2.1 – Comparaison des modèles existants

2.3 Contrôle d'accès aux services Web

2.3.1 XACML

XACML [41] est un standard de l'OASIS [3] qui offre un langage, des concepts et un environnement pour la création et la gestion des politiques de contrôle d'accès pour les services Web et les applications distribuées. L'objectif principal de XACML est de pouvoir générer une décision (accord ou refus) face à une demande d'accès à une ressource. XACML se base sur XML, donc il est indépendant des plateformes matérielles et logicielles. De plus, il permet de rendre le contrôle d'accès modulaire et réutilisable.

2.3.1.1 Architecture de XACML

Le langage XACML a apporté une architecture et des concepts qui fournissent les grandes lignes pour concevoir un système de contrôle d'accès. Cette architecture vise à atteindre plusieurs objectifs :

- Assurer une protection efficace des ressources, et ce, du point de vue du contrôle d'accès.
- Permettre de concevoir un système indépendant de la plate-forme utilisée.
- Permettre d'intégrer le système de contrôle d'accès dans des applications déjà existantes.

L'architecture globale du langage consiste en plusieurs composantes collaborant entre elles. On présentera les différentes composantes dans la suite de cette section.

1. **Point d'administration des politiques** Le point d'administration ou PAP pour Policy Administration Point, est l'entité qui crée les règles, les politiques et les ensembles de politiques de contrôle d'accès.
2. **Point d'application des politiques** Le point d'application des politiques ou PEP (Policy Enforcement Point) est l'entité qui protège les ressources. Le PEP interagit avec des entités extérieures (applications ou utilisateurs) via des requêtes d'accès.
3. **Point de décision des politiques** Le centre de décision des politiques ou PDP (Policy Decision Point) est l'entité en charge de sélectionner les règles, les politiques ou les ensembles de politiques qui sont applicables à une requête donnée. Il évalue les cibles et les conditions afin d'aboutir à une décision.
4. **Source d'information de politique** La source d'information de politique ou PIP (Policy Information Point) a pour rôle d'extraire les informations supplémentaires qui ne sont pas présentes dans la demande d'accès.

La figure 2.8 illustre les étapes de traitement d'une requête.

3. Organization for the Advancement of Structured Information Standards : consortium mondial qui travaille pour la standardisation de formats de fichiers ouverts basés notamment sur XML

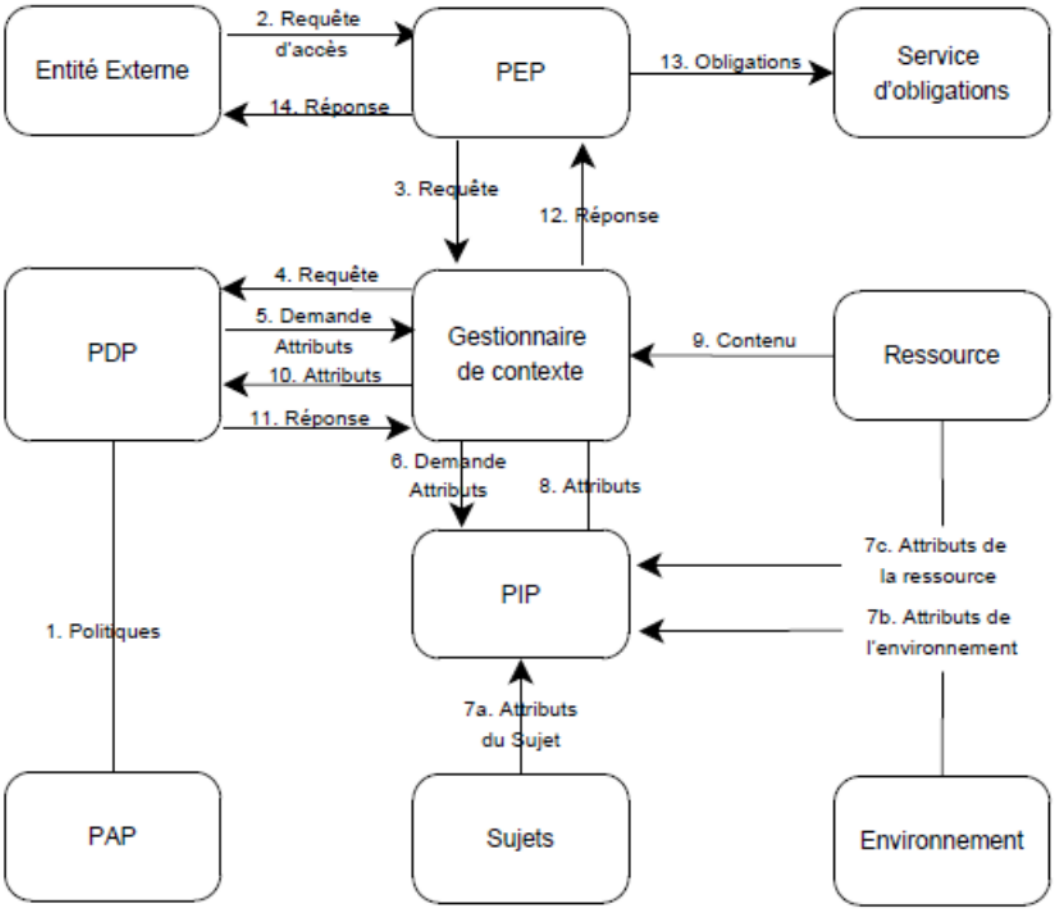

FIGURE 2.8 – Le flux de données dans un environnement XACML

2.3.1.2 WS-XACML (Web Service XACML)

WS-XACML [4] spécifie comment utiliser XACML dans l'environnement des services Web. WS-XACML présente deux nouveaux types de politique d'assertions pour permettre à des fournisseurs de service Web et aux clients de spécifier leur autorisation, les conditions et la possibilité d'interactions entre les services Web.

2.3.1.3 Avantage

- C'est un standard ouvert : En utilisant un langage standardisé et disponible à tous (aussi bien aux experts qu'aux utilisateurs), il n'est pas nécessaire de développer un produit propre ou de dépendre d'un autre fournisseur.
- Il est générique : En conservant un certain niveau d'abstraction, il n'est pas dépendant d'une implémentation particulière ou d'un type de système particulier.
- Il est distribué : Une politique peut faire référence à d'autres politiques, au besoin à des politiques distantes ; ce qui permet de ne pas devoir gérer les règles de manière monolithique et facilite l'interopérabilité en garantissant que toutes les politiques étendant une même source seront compatibles.
- Il est puissant : En plus de la facilité d'extension, la plupart des constructions

nécessaires aux utilisations courantes sont déjà disponibles, ce qui facilite encore son déploiement.

2.3.1.4 Inconvénient

- L'administration est centralisée, ce qui présente un point vulnérable aux attaques ;
- Il ne supporte pas la négociation ;
- Il n'est pas applicable pour les services Web composites ;
- Il n'utilise pas de sémantique.

2.3.2 X-RBAC (Xml Role Bases Access Control)

Le langage de spécification pour X-RBAC [19] vise à modéliser les éléments de base RBAC et leurs associer un ensemble de relations. Pour représenter les éléments RBAC en XML, des schémas de définitions sont générés pour "utilisateur", "rôle" et "autorisation".

2.3.2.1 Processeur XML :

Le processeur XML contient l'analyseur XML et les représentations arbre DOM des documents XML, Le système X-RBAC fournit un chargeur de politique qui charge les feuilles de la politique pour une politique donnée. L'étape suivante, la fonctionnalité est fournie pour valider les feuilles de la politique en matière de vérification de l'existence et la conformité du type. Cela signifie que tous les utilisateurs, les rôles et les autorisations mentionnées dans XURAS et XPRAS doivent exister dans XUS, XRS et XPS respectivement correspondant. En outre, les types d'informations d'identification associés aux utilisateurs doivent être conformes aux définitions de type dans la feuille de XCredTypeDef. Une fois les feuilles de la politique sont validées, la représentation de l'arbre DOM correspondant est générée et envoyée sur le processeur RBAC.

2.3.2.2 Processeur RBAC :

Le processeur RBAC contient le module RBAC ainsi que les portions de codes généré par le module RBAC. Il effectue les tâches d'administration et l'exécution des politiques.

2.3.2.3 Avantages

- Il offre les avantages du modèle RBAC pour les services Web ;
- La propagation des permissions d'accès permet de réduire le nombre de permissions à affecter.

2.3.2.4 Inconvénients

- L'administration est centralisée, ce qui présente un point vulnérable aux attaques ;

- Il ne supporte pas les conditions et les contextes ;
- Il ne supporte pas la négociation ;
- Il n'est pas applicable pour les services Web composites ;
- Il n'utilise pas de sémantique.

2.3.3 Role Based Access Control for Web Services

Ce modèle [88] présente une approche sur le contrôle d'accès aux services Web, il supporte aussi les politiques de contrôle d'accès pour les services Web composites. Il est constitué d'une adaptation du contrôle d'accès basé sur les rôles. Pour les services Web où les sujets sont des services Web et les objets sont aussi des services Web et leurs attributs, les autorisations sont définies sur les deux services Web (objet et sujet), leurs attributs et les services Web composites.
Le modèle est divisé en deux parties, la première est une adaptation de RBAC pour les services Web simples et la deuxième est une extension pour les services Web composites.

2.3.3.1 Le modèle RBAC pour les services Web simples (SWS-RBAC)

Ce modèle hérite du RBAC standard (en particulier la hiérarchie des rôles). Les autorisations sont exprimées en deux niveaux. Le niveau service et le niveau attribut. Un sujet peut accéder à un attribut d'un service (objet) seulement si son rôle a l'autorisation d'utiliser le service et il a les privilèges d'accès exigés sur l'attribut donné du service. La figure 2.9 représente les éléments du modèle SWS-RBAC et les relations. Chaque utilisateur est associé à un ou plusieurs rôles, un rôle accorde une autorisation d'accès à un ou plusieurs services et un mode d'accès est appliqué à un ou plusieurs attributs, un mode d'accès peut se composer d'un ou plusieurs modes d'accès spécifique. Un service au minimum un mode d'accès appliqué pour un ou plusieurs attributs. Afin d'exécuter un service, un rôle doit passer le contrôle d'accès au niveau service, et au niveau attribut, sinon, il doit avoir l'accord d'exécuter le service et le mode d'accès sur les attributs requis.

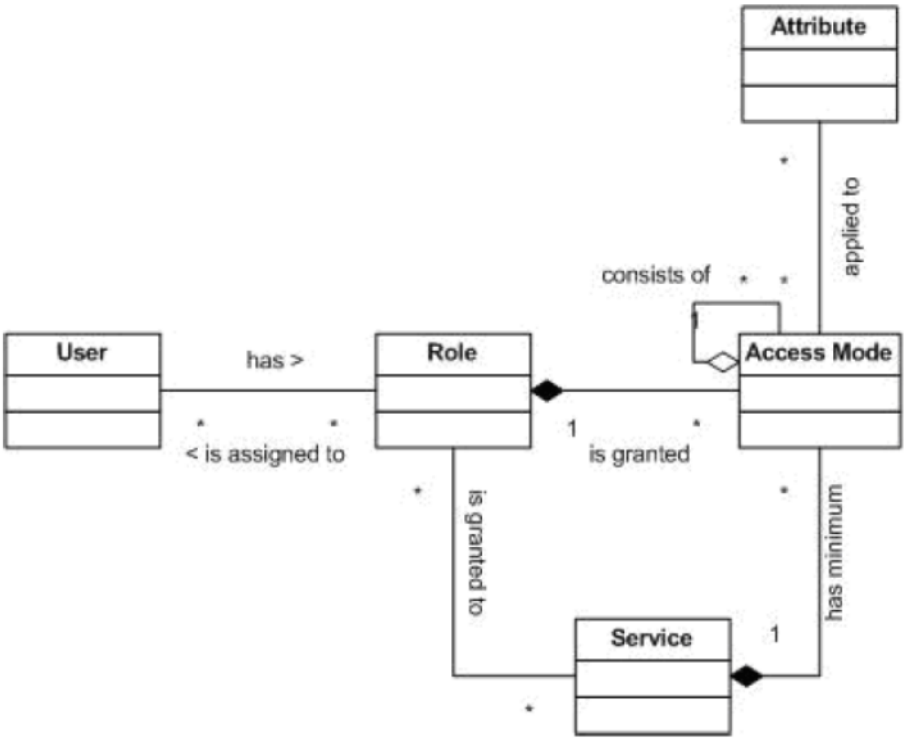

FIGURE 2.9 – Architecture de SWS-RBAC [88]

2.3.3.2 Le modèle RBAC pour les services Web composites (CWS-RBAC)

Quand un utilisateur invoque un service Web composite, il utilise ses privilèges pour appliquer ses tâches. Ainsi, quand un service composite appelle un service simple d'un autre fournisseur, il utilise le rôle global de l'utilisateur utilisé pour invoquer le service composite aux rôles locaux sur chaque fournisseur. La figure 2.10 présente les différents composants de CWS-RBAC.

2.3.3.3 Avantages

- L'administration distribuée réduit sa vulnérabilité aux attaques ;
- Il est applicable aux services Web composites ;
- Il est flexible.

2.3.3.4 Inconvénients

- Il ne supporte pas les conditions et les contextes ;
- Il ne supporte pas la négociation ;
- Il n'utilise pas de sémantique ;
- Le concept du rôle global le rend gourmand en espace mémoire et temps.

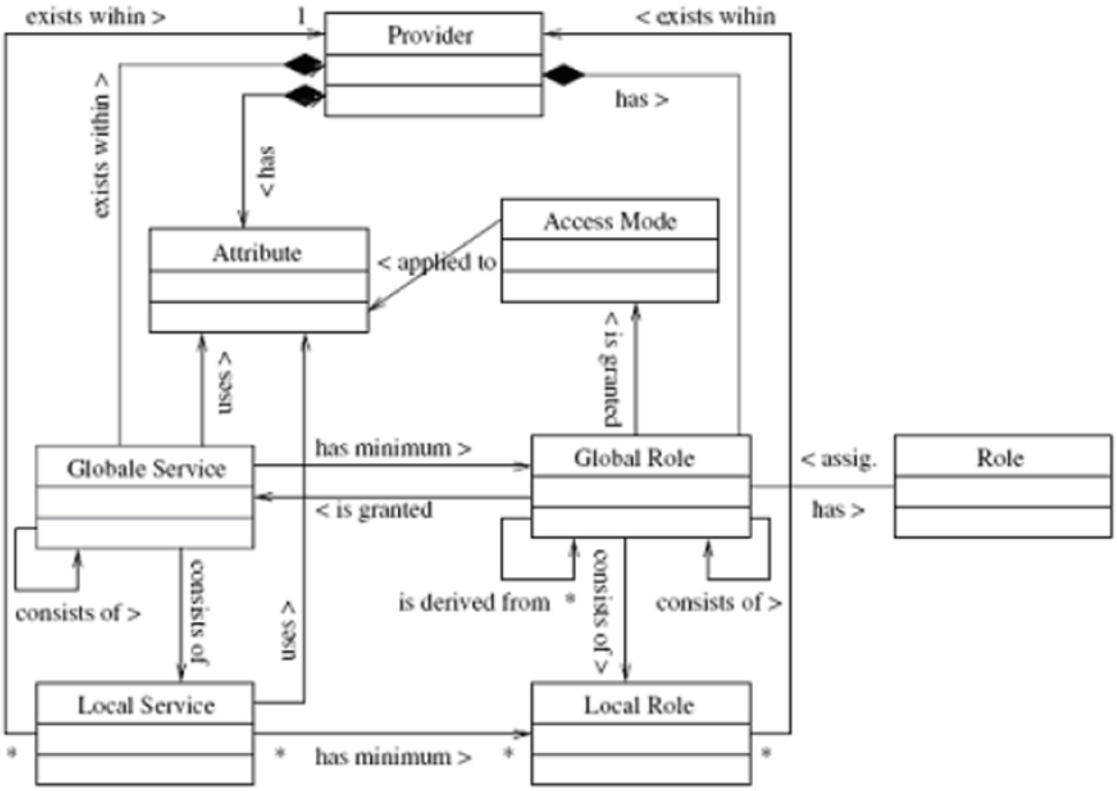

FIGURE 2.10 – Architecture de CWS-RBAC [88]

2.3.4 WS-AC (Web Service Access control)

WS-AC [16] se compose d'une approche innovatrice pour fournir un mécanisme de contrôle d'accès dans un environnement distribué faiblement couplé tel que le service Web. WS-AC est un modèle de contrôle d'accès au document XML, il laisse exprimer, valider et imposer des politiques de contrôle d'accès sans supposer de confiance préétablie entre les utilisateurs du service Web, les conditions d'accès sont exprimées en terme des attributs de l'utilisateur et des paramètres qui caractérisent le service Web. Il est possible de spécifier qu'un utilisateur peut utiliser un service Web donné, mais seulement avec des valeurs spécifiques des paramètres du service Web. En plus, ce modèle supporte la négociation d'accès entre l'utilisateur et le fournisseur du service, les concepts de négociation rendent ce modèle de contrôle d'accès capable de demander aux utilisateurs de raffiner leurs requêtes d'accès (changer les paramètres du service). Enfin, la fidélité des utilisateurs qui est une condition préalable pour l'exactitude de ce modèle est réalisée en incluant des attributs certificats avec des demandes d'accès.

Les politiques de contrôle d'accès sont spécifiées en contraignant les paramètres du service et les attributs que les sujets devraient posséder pour obtenir un service. Quand une requête est effectuée, le système vérifie les politiques de contrôle d'accès correspondantes, pour vérifier si la demande peut être acceptée comme elle est, doit être rejetée ou doit

être négociée.
La négociation consiste à éliminer ou à modifier quelques paramètres de service. WS-AC1 [16, 15] est une version de WS-AC implémenté pour les services Web sans état [4].

2.3.4.1 Processus de WS-AC

La figure 2.11 illustre le processus du WS-AC :

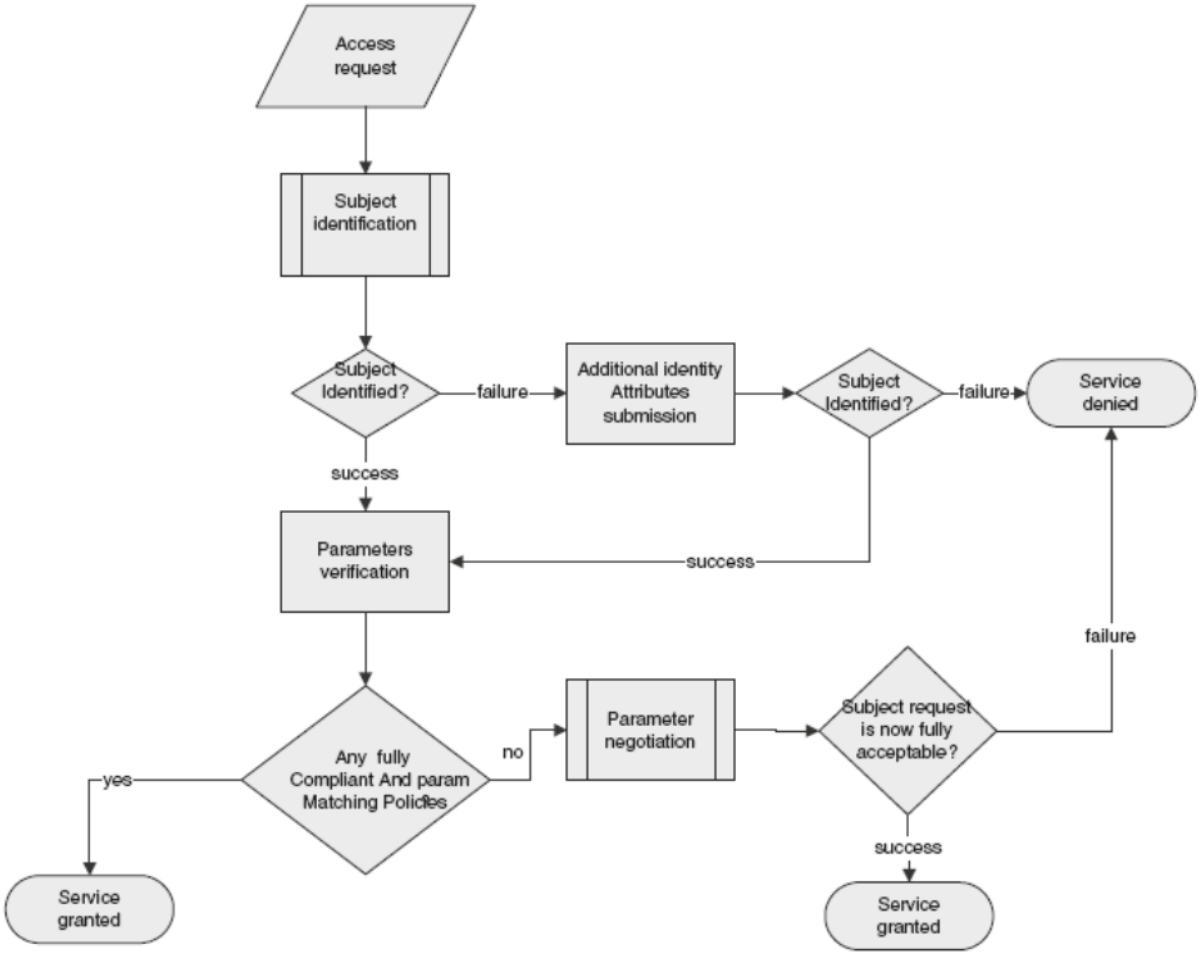

FIGURE 2.11 – WS-AC1 [16]

2.3.4.2 Avantages

- Le concept de négociation rend le modèle flexible ;
- Supporte les conditions des attributs du sujet ;

2.3.4.3 Inconvénients

- L'administration est centralisée, ce qui présente un point vulnérable aux attaques ;
- Il n'est pas applicable pour les services Web composites ;

4. Un service Web sans état est un service Web qui ne garde aucune trace des appels

2.3.5 X-GTRBAC

X-GTRBAC [18] est une extension du modèle de contrôle d'accès contextuel GTRBAC. Ainsi son administration repose sur des permissions administratives telles que : assign, deassign, assignp, deassignp, enable, disable, map, unmap. X-GTRBAC inclut un modèle dont le but est de gérer une politique d'administration liée à une grande entreprise. Pour cela, il fournit un mécanisme qui distribue l'autorité administrative sur plusieurs domaines de l'entreprise. La portée du domaine permet de restreindre les droits des administrateurs.
X-GTRBAC est une adaptation du modèle GTRBAC sur XML pour supporter l'application de politique dans un environnement hétérogène et distribué. GTRBAC prolonge le modèle de contrôle d'accès basé sur les rôles RBAC. RBAC emploie le concept des rôles pour représenter un ensemble de permissions dans une organisation.

2.3.5.1 Architecture de X-GTRBAC

La figure 2.12 présente l'architecture de X-GTRBAC :

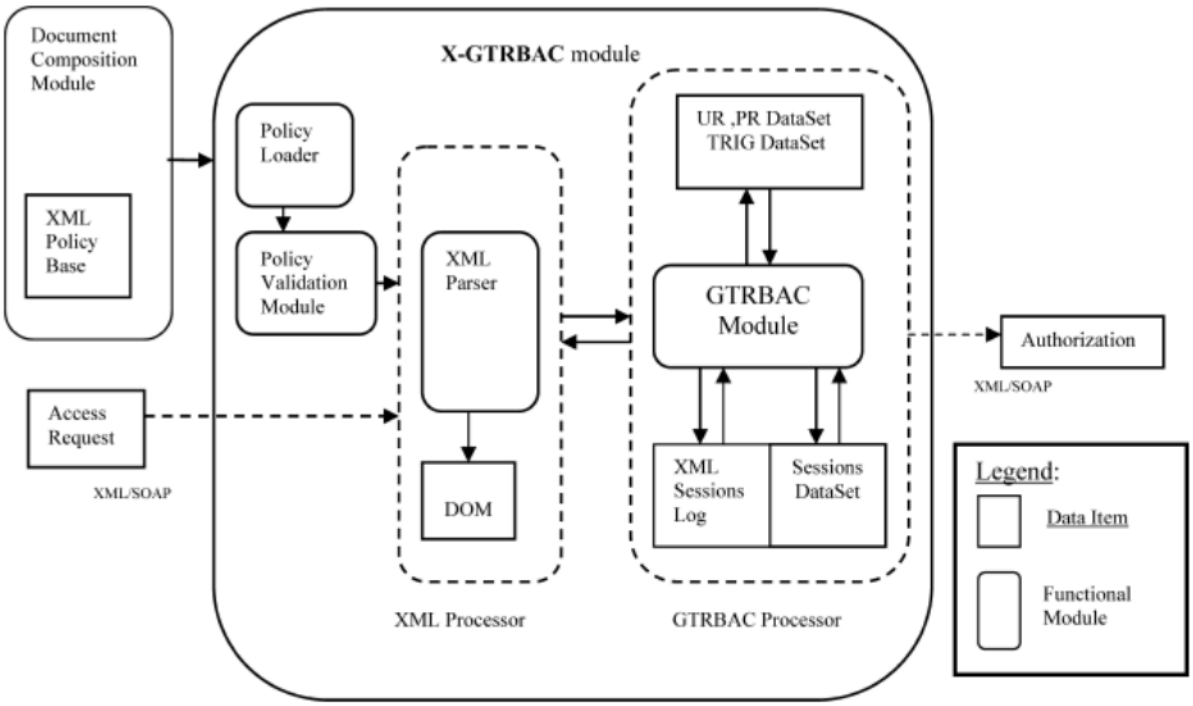

FIGURE 2.12 – Architecture de X-GTRBAC [18]

Comme indiqué dans la figure 2.12, les deux sous-systèmes du module X-GTRBAC sont le processeur GTRBAC et le processeur XML.

Processeur XML : Le processeur XML a un fonctionnement similaire à celui du processeur XML du modèle X-RBAC

Processeur GTRBAC : Le processeur GTRBAC contient le module GTRBAC ainsi que les portions de codes générés par le module GTRBAC. Il effectue les tâches d'administration et l'exécution des politiques.

2.3.5.2 Avantages

- Il offre les avantages du modèle GTRBAC pour les services Web ;
- La propagation des permissions d'accès permet de réduire le nombre de permissions à affecter.

2.3.5.3 Inconvénients

- L'administration est centralisée, ce qui présente un point vulnérable aux attaques ;
- Il ne supporte pas la négociation ;
- Il n'est pas applicable pour les services Web composites ;
- Il n'utilise pas de sémantique.

2.3.6 Comparaison des modèles de contrôle d'accès pour les services Web

Dans cette section nous réalisons une comparaison entre les modèles de contrôle d'accès aux services Web qu'on a présenté. Avant d'entamer la comparaison, nous présenterons les différents critères que nous jugeons important.

- **Granularité :** Représente la manière dont les objets sont définis dans le modèle de contrôle d'accès. L'objet peut être vu comme attribut/élément XML, des instances de documents, des ensembles de documents ou un service Web ;
- **Support du rôle :** Indique que le modèle de contrôle d'accès utilise le concept de rôle ;
- **Administration :** Représente la manière dont le modèle de contrôle d'accès est administré ;
- **Conditions :** Représente les conditions que le modèle de contrôle d'accès prend en compte tel que le temps, localisation, profil utilisateur... etc.
- **Obligations :** Les obligations sont des fonctions qui s'exécutent lors d'une décision de contrôle d'accès ;
- **Propagation :** Représente les politiques de propagation utilisées par le modèle de contrôle d'accès.
- **Négociation :** Représente la faculté du modèle de contrôle d'accès de permettre au sujet d'adapter ses requêtes.
- **Extensibilité :** Représente la capacité du modèle de permettre à l'utilisateur d'exprimer des autorisations utilisant des fonctions et des conditions qui ne sont

pas déjà prédéfinies.

- **Flexibilité :** Représente la capacité du modèle de contrôle d'accès à être flexible.
- **Sémantique :** Représente la capacité du modèle de contrôle d'accès à donner un sens aux requêtes et aux permissions d'accès.
- **Composition :** Représente la capacité du modèle de contrôle d'accès à supporter des requêtes destinées à un service Web composite.

Le tableau 2.2 illustre une comparaison entre les modèles de contrôle d'accès aux services Web.

Modèle	**XACML**	**WS-AC1**	**WS-RBAC**	**X-RBAC**	**X-GTRBAC**
Granularité	-Éléments -Attribut	-Éléments - Attributs	-Attributs de service Web simple -service Web composite	-Élément -Attribut -Instance d'un document -Schéma	-Élément -Attribut -Instance d'un document -Schéma
Administration	Centralisée	Centralisée	Distribuée	Centralisée	Centralisée
Support du rôle	Supporté	non supporté	-rôle local - rôle globale	Supporté	Supporté
Support de conditions	Supporté	Supporté	non supporté	Non supporté	Supporté
Obligation	Supportée	Non supportée	Non supportée	Non supportée	Non supportée
Propagation	Non supportée	Non supportée	Non supportée	Propagation des permissions	Propagation des permissions
Négociation	Non supportée	Supportée	Non supportée	Non supportée	Non supportée
Extensibilité	Oui	Non	Non	Non	Non
Sémantique	Pour le matching	Non	Non	Non	Non
Composition	Non	Non	Pour le modèle CSW-RBAC	Non	Non
Flexibilité	Flexible	Flexible	Flexible	Restrictif	Restrictif

TABLE 2.2 – Comparaison des modèles de contrôle d'accès aux services Web

2.4 Conclusion

Un modèle de contrôle d'accès protège les services Web à partir des opérations des clients qui ne satisfont pas les exigences citées.

Dans ce chapitre, nous avons présenté en premier les modèles de contrôles d'accès aux systèmes d'informations. Par la suite nous avons présenté les modèles de contrôle d'accès aux services Web les plus connus dans la littérature. Pour enfin faire une comparaison entre les différents modèles de contrôle d'accès aux services Web.

La différence entre les modèles de contrôle d'accès aux systèmes d'informations et le contrôle d'accès aux services Web réside dans le nombre d'utilisateurs gérés, les protocoles de communications utilisés et l'accessibilité aux ressources.

Le chapitre suivant présente les modèles formels pour le contrôle d'accès.

Chapitre 3

Spécification formelle des modèles de contrôle d'accès

3.1 Introduction

En informatique, les méthodes formelles sont utilisées pour renforcer la robustesse des systèmes. Ce sont des techniques permettant de raisonner rigoureusement. Dans les méthodes formelles, toutes les fonctions sont décrites principalement par des notions mathématiques. Or les notions mathématiques impliquent nécessairement des preuves mathématiques. C'est ce qui fait la différence avec la méthode traditionnelle, qui ne peut pas prouver ce qu'elle décrit, car elle ne peut que le tester.
Une méthode de spécification est dite formelle lorsqu'elle utilise un ou plusieurs langages de spécification formels.
Les éléments clés d'une méthode formelle sont :

- Langage formel pour l'écriture de spécifications,
- Règles pour évaluer la validité/qualité des spécifications,
- Stratégies et règles pour raffiner (mettre en œuvre) les spécifications et vérifier ces raffinements.

3.2 Langage formel

Un langage de spécification est dit formel lorsqu'il obéit à une syntaxe formelle stricte, servant à exposer des énoncés de manière précise, si possible concise et sans ambiguïté. Ses notations sont rigoureusement définies par une sémantique précise.
Les langages formels sont étudiés principalement dans les domaines des mathématiques, de la logique, de l'informatique et de la linguistique.

En mathématiques, en logique et en informatique, un langage formel est formé :

- D'un ensemble de mots obéissant à des règles logiques strictes (dites grammaire formelle ou syntaxe).
- D'une sémantique sous-jacente.

La force des langages formels est de pouvoir faire abstraction de la sémantique, ce qui rend les théories réutilisables dans plusieurs modèles.
Un langage formel peut être spécifié par différents moyens, comme :

- Les mots produits par des règles de production dites aussi règles d'une grammaire formelle,
- Les mots générés par une expression rationnelle,
- Les mots acceptés par un certain automate, comme une machine de Turing ou un automate d'états finis,
- L'ensemble des instances d'un problème de décision, dont la réponse est OUI,
- Des résultats d'inférences.

3.3 Travaux sur la formalisation des modèles de contrôle d'accès

3.3.1 Un langage logique pour l'expression des autorisations

Les auteurs [47] proposent un modèle de contrôle d'accès capable de supporter différentes politiques de contrôle d'accès pour les systèmes d'information et un langage basé sur la logique des prédicats pour la spécification des autorisations d'accès du modèle proposé. Les modèles de contrôle d'accès peuvent être représentés par ce modèle par le langage logique en spécifiant les différentes entités, les autorisations d'accès et la politique de contrôle d'accès. Les politiques sont exprimées par des règles qui appliquent la dérivation des autorisations, la résolution des conflits, le contrôle d'accès et la vérification des contraintes d'intégrité.

3.3.1.1 Concepts

Dans [47], les concepts utilisés sont :

Objet : Noté (obj), objet est l'ensemble des entités définis dans le système sur lequel certaines actions peuvent être exécutées. Les objets sont regroupés en type (T) dont chaque type est spécifique à une application donnée.

Action : les actions sont définies selon le type (T), elles différent d'une application à une autre (exemple pour une application de base de données, les actions sont (select, update, insert... etc.)).

Sujet : les autorisations peuvent être accordées à trois types de sujets (utilisateur, rôle, groupe). L'utilisateur est un individu qui se connecte au système et qui émet des requêtes. Le groupe est un ensemble d'utilisateurs, les groupes peuvent être imbriqués. Les rôles sont des collections de privilèges nécessaires pour effectuer des activités spécifiques dans le système.

Autorisation : Une autorisation est un quadruple (u, o, R, a) qui représente respectivement l'utilisateur, l'objet, le rôle, et l'action. Les actions sont précédées par un signe (+, -) qui signifie respectivement (accepté, refusé).

3.3.1.2 Modélisation par la logique des prédicats

Symboles constants : Les symboles constants sont $(Obj \cup T \cup U \cup G \cup A \cup SA \cup N)$ ou (N) est l'ensemble d'entiers. (Obj) est l'ensemble d'objets,(T) l'ensemble de types, (U) l'ensemble d'utilisateurs, (G) l'ensemble de groupes, (R) l'ensemble de rôles, et (A) et (SA) l'ensemble d'autorisations signées ou non signées.

Les symboles variables : Les ensembles, $V_o, V_t, V_u, V_g, V_r, V_R, V_a, V_sa$ représentent les symboles variables sur les ensembles respectivement $obj, T, U, G, R, 2^R, A$ et SA.

Les symboles prédicats : Les symboles prédicats sont représentés comme suit :

- **cando** c'est un symbole prédicat à trois arguments, le premier argument est un objet, le deuxième est un sujet et le troisième est un signe d'autorisation. cando représente les autorisations explicites insérées par le SSO (the Système Sécurity Office).
- **dercando** a les mêmes arguments que cando, il représente les autorisations dérivées par le système en utilisant les règles logiques d'inférence.
- **do** avec les mêmes arguments que cando, il représente les autorisations qui sont affectées pour chaque sujet à chaque objet. Il met en œuvre la politique de résolution de conflit.
- **grant** contient quatre arguments dont le premier argument est l'objet, le deuxième argument est l'utilisateur, le troisième l'ensemble des rôles, le quatrième est le signe de l'action. Il applique la politique de contrôle d'accès.
- **done** contient cinq arguments qui représentent l'objet, le sujet, le rôle, terme d'action non signé et un nombre naturel. done représente les actions exécutées par un utilisateur.

- **active** dont le premier argument représente l'utilisateur et le deuxième représente le rôle. Il représente l'activation d'un rôle pour un utilisateur.
- **dirin et in** ils contient les relations des membres directs et indirects entre les sujets.
- **typeof** qui prend pour arguments un objet o et un type t, il donne la relation de regroupement entre les objets.

La spécification des autorisations : La spécification des autorisations est représentée par les différents symboles définis.
Une spécification d'autorisation est un ensemble de règles dont l'évaluation détermine, pour chaque demande d'accès qui peut être soumise, si l'accès demandé doit être satisfait ou refusé. Formellement une spécification d'autorisation est un ensemble de règles d'autorisation (cando), de dérivation (dercando), de résolution (do), de contrôle d'accès (grant) et d'intégrité (error). Une spécification d'autorisation est complète si pour chaque quintuple (o, u, R, a) au moins un de grant $(o, u, R, +a)$ et de grant $(o, u, R, -a)$ est vrai. Une spécification d'autorisation est cohérente si pour chaque quintuple (o, u, R, a) grant $(o, u, R, +a)$ et de grant $(o, u, R, -a)$ ne sont pas vraies au même temps.

Exemple : Dans cet exemple, les auteurs représentent un modèle de contrôle d'accès existant (The Sea View Verification [87]) avec le modèle proposé.
Dans le modèle [87], les sujets peuvent être des utilisateurs ou des groupes. Les groupes sont des ensembles d'utilisateurs individuels et ne peuvent pas être imbriqués. Les autorisations négatives ne sont pas prises en considération, un privilège nul (signifie qu'aucun accès n'est autorisé) est utilisé à la place. Un utilisateur peut exercer les privilèges d'un seul groupe à un moment donné (les groupes sont comme les rôles) un sujet demandeur est le couple formé de l'utilisateur et du groupe de l'utilisateur à activer. La demande d'un sujet à exercer une action donnée sur un objet n'est accordée que si l'une des conditions suivantes est satisfaite : (i) l'utilisateur a l'autorisation pour l'accès et il n'a pas d'autorisation nulle pour cette action, ou (ii) l'utilisateur ne dispose d'aucune autorisation sur l'objet et le groupe a l'autorisation pour l'accès et n'a pas d'autorisation nulle sur l'objet. En raison de la façon dont les groupes sont utilisés, ils peuvent être représentés comme des rôles dans le ce modèle. Ainsi, une permission dans [87] peut être exprimé comme suit :

$do(o, u, s, +a) \leftarrow cando(o, u, +a)\ \&\ \neg cando(o, s, +null).$
$grant(o, u, R, +a) \leftarrow do(o, u, +a).$
$grant(o, u, R, +a) \leftarrow \neg cando(o, u, +a')\ \&\ active(u, r)\ \&\ do(o, r, R, +a).$
$grant(o, u, R, -a) \leftarrow \neg grant(o, u, R, +a).$
$error() \leftarrow int(s, s')\ \&\ s \neq s'.$
$error() \leftarrow do(o, u, s, -a).$
$error() \leftarrow active(u, r)\ \&\ active(u, r')\&\ r \neq s'.$

3.3.1.3 Avantages

- Le modèle proposé peut supporter les concepts de rôle et de groupe ;
- Le modèle proposé peut être utilisé afin de spécifier les différentes politiques de contrôle d'accès discrétionnaires existantes.

3.3.1.4 Inconvénients

- Le modèle proposée ne peut pas exprimer les modèles qui sont sensibles aux contraintes ;
- Le modèle est plus adapté pour les politiques de contrôle d'accès discrétionnaires. En effet, l'ensemble de prédicats qui peuvent être utilisés dans ce modèle est fixe et conçu spécifiquement pour exprimer les modèles de contrôle d'accès discrétionnaires traditionnels.
- Le formalisme de la logique des prédicats est compréhensible que par les machines et il est difficile à être interpréter par un humain.

3.3.2 Un cadre logique pour le raisonnement sur les modèles de contrôle d'accès

Dans l'article [14], les auteurs proposent un cadre formel pour raisonner sur les modèles de contrôle d'accès. Le cadre proposé est basé sur un formalisme logique et a été développé dans le but d'être aussi générale que possible contrairement au modèle de [47] qui est plus adapté aux politiques de contrôle d'accès discrétionnaire. Par conséquent, il contient un ensemble de primitives nécessaires sur lequel tous les autres concepts peuvent être construits. Puis, selon le modèle représenté, certains de ces blocs de construction sont sélectionnés et composés ensemble. Chaque instance du cadre proposé correspond à un programme C-Datalog, interprété selon la sémantique des modèles.

3.3.2.1 Concepts

Le modèle proposé prend en charge la représentation de quatre concepts de base, le sujet, l'objet, les privilèges et la session.

Sujets : les autorisations d'accès sont attribuées au sujet, le sujet peut être un utilisateur, un processus, un groupe ou un rôle. Un groupe est un ensemble d'utilisateurs et peut être représenté en hiérarchie. Un rôle représente une fonction dans une organisation donnée. Les rôles peuvent être représentés sous forme d'une hiérarchie. Un processus est l'exécution d'un programme pour le compte d'un utilisateur.

Objets : Les objets sont les ressources à protéger. Les objets peuvent être organisés hiérarchiquement.

Privilèges : Les privilèges représentent les modes d'accès que les sujets peuvent exercer sur les objets dans le système. Certaines interactions peuvent exister entre les privilèges, en précisant qu'un privilège est plus fort que d'autres. Pour cette raison, les droits peuvent également être organisés sous forme d'une hiérarchie des privilèges.

Session : Une session est une instance d'une connexion d'un utilisateur au système.

Règle d'autorisation : Les autorisations sur les éléments de base peuvent être spécifiées par les règles d'autorisation. Les règles d'autorisation peuvent exploiter les sujets, les objets, les privilèges et les sessions pour la dérivation des autorisations positives ou négatives. La spécification des règles d'autorisation peut s'appuyer sur l'utilisation des prédicats définis par l'utilisateur.

Règles de contrainte : Les contraintes sur les composants du système peuvent être spécifiées par des règles de contrainte. En général, les règles de contrainte spécifient les conditions qui ne doivent pas être violées par les composantes du système. Les contraintes peuvent être classées en contrainte statique et dynamique : les contraintes statiques peuvent être vérifiées de manière statique, c'est-à-dire sans prendre compte de l'état d'exécution du système, alors que les contraintes dynamiques ne peuvent être vérifiées qu'en prenant compte de l'état d'exécution du système. Les contraintes peuvent concerner les sujets, les objets, les privilèges, et les sessions.

3.3.2.2 Modélisation

Pour représenter formellement les blocs d'instructions du cadre, les auteurs ont utilisé CDatalog [71], qui est une extension orientée objet de Datalog, supportant ainsi toutes les caractéristiques requises pour modéliser le sujet, l'objet, les privilèges et la session.

Représentation des composantes de base de C datalog Afin de représenter les différents composants en C-Datalog, un CD [1] Schéma doit être défini en premier, le CD Schéma peut définir les composants suivants :

- **Composante domaine (DC) :** Les classes 'domaine' représentent la structure des composants de base du cadre (sujets, objets, privilèges, et sessions) et les instances du domaine représentent les sujets actuels, les objets, les privilèges, et les sessions. Les instances sont représentées sous forme d'un ensemble de faits, qui forment le composant domaine.
- **Composant structure du domaine (DSC) :** Informations sur la structure domaine représente les relations existantes entre les composantes de base, par exemple relations père-fils dans les hiérarchies de composants de base. Cette information est

1. C-Datalog

exprimée à travers un ensemble de règles de relations dérivées, ce qui représente la composante structure du domaine.

- **Composant autorisation (CA) :** Ce composant contient des règles d'autorisation, exprimées en un ensemble de règles de relations dérivées.
- **Composante propagation (PC) :** Cette composante consiste à dériver des règles, par lequel des autorisations supplémentaires peuvent être obtenues, à partir de règles d'autorisation et de l'information du domaine.
- **Composante de contrainte (CC) :** La composante de contrainte est composée de règles dérivées capables d'exprimer des contraintes statiques et dynamiques sur les composants de base.

un Schéma de modèle de contrôle d'accès 'CD Schéma' est représenté comme suit : $< B, Scheme, A, ISA, Z >$ tel que :

1. $B = K \cup R$ tel que : (i) $K = K_{built_in} \cup K_{basic}$ ou $K = K_{built_in} = \perp, intstring$ and $K = K_{built_in} \subseteq \{user, group,\ role, process, subject, object, privilege, session\}$ en s'assurant que K_{basic} contient toujours le nom de classe "sujet","objet", et au moins un nom de classe de { utilisateur, groupe, rôle et processus } ; (ii)$R = R_{domaien} \cup R_{auth} \cup R_{contraint} \cup R_{user_{def}}$. En particulier $R_{domaine} \subseteq \{SubG, InSubG, Belong, UserIn, LessR, InLessR, Play, UserPlay, PartOf, InPertOf, LessP, InLessP, ActvieRole\}$, $R_{auth} \subseteq \{Auth_d, Auth_p, Auth\}, R_{constraint} \subseteq \{ErrorC\}$,et $R_{user_{def}}$ est un ensemble d'utilisateur.K et R_{domain} sont liés comme suit :

   - Si $SubG \in R_{domain}$, alors $Group \in K_{basicet}$ les groupes doivent être organisés en hiérarchie $(InSubG \in R_{domain})$;
   - Si $Belong \in R_{domain}$, alors $\{user, group\} \subseteq K_{basic}$;
   - Si $UserIn \in R_{domain}$, alors $\{user, group\} \subseteq K_{basic}, Belong \in R_{domain}$ et les groupes doivent être organiser en hiérarchie $(InSubG \in R_{domain})$;
   - Si $Less \in R_{domain}$, alors $Role \in K_{basic}$ et les rôles doivent être organisés en hiérarchie $(InLessR \in R_{domain})$;
   - Si $Play \in R_{domain}$, alors $user, role \subseteq K_{basic}$;
   - Si $UserPlay \in R_{domain}$, alors $user, role \subseteq K_{basic}, Play \in R_{domain}$ et les rôles doivent être organisés en hiérarchie $(InLessR \in R_{domain})$;
   - Si $PartOf \in R_{domain}$, alors $object \in K_{basic}$ et les objets doivent être organisés en hiérarchie $(InPartOfinR_{domain})$;
   - Si $LessP \in R_{domain}$, alors $Privilege \in K_{basic}$ et les privilèges doivent être organisés en hiérarchie $(InLessP \in R_{domain})$;
   - Si $ActiveRole \in R_{domain}$, alors $\{user, role, session\} \subseteq K_{basic}$.

2. Schéma représente le Schéma de fonction, pour les noms de relation dérivées ;

3. Fonction ISA, pour les noms de classe non Intégrés, elles sont définies comme suit : $0I1SA(user) = ISA(group) = ISA(role) = ISA(process) = subject$ (voire la figure 3.1) pour le reste $ISA(c) = \emptyset$;
4. Le nom d'attribut de l'ensemble A contient le nom « Self » et le nom d'attribut qui apparait dans le schéma de chaque nom d'entité $b \in B$;
5. Un ensemble Z représente les oids [2] (Object Identifier) les noms d'entités de classe.

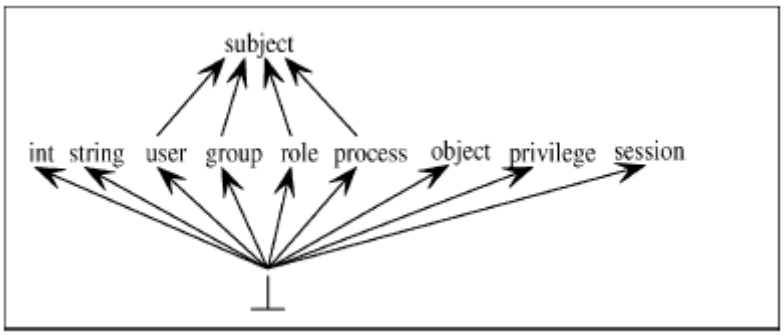

FIGURE 3.1 – ISA hiérarchie

Le tableau 3.1 illustre quelques exemples de prédicats du modèle.

Catégorie	**Prédicat**	**Schéma de fonction**	**Signification**
Groupe	$SubG$	$SubG(G_1 : Group, G_2 : Groupe)$	Représente la relation père-fils dans la hiérarchie des Groupe
	$InSubG$	$InSubG(G_1 : Group, G_2 : Groupe)$	Représente la fermeture transitive de $SubG$; G_1est un fils indirecte de G_2
	$Belong$	$Belong(U : user, G : Groupe)$	Représente l'appartenance de l'utilisateur u dans le Groupe G
	$UserIn$	$UserIn(U : user, G : Group)$	Représente la fermeture transitive de $Belong$; utilisateur u appartient indirectement au groupe G

TABLE 3.1 – Prédicats du modèle

Le tableau 3.2 représente quelques exemples de modélisation de règles et leurs significations.

2. (Object IDentifier) sont des identifiants universels, Ils ont été définis dans une recommandation de l'International Telecommunication Union, La RFC 2256 normalise un certain nombre de ces objets.

Règle	**Signification**
$ErrorC \leftarrow Belong(U : X, G : Y)$	L'utilisateur X ne peut pas être membre du groupe Y
$ErrorC \leftarrow Belong(U : X, G : Y), Belong(U : Y, G : K), belong(U : Z, G : K)$	Les utilisateurs X, Y, Z ne peuvent pas être dans le même groupe K
$ErrorC \leftarrow Active(U : X, S : Y, R : Z), ActiveRole(U : X, S : Y, R : K)$	Les rôles Z, K ne peuvent pas être affecter à l'utilisateur X avec la même session
$ErrorC \leftarrow Auth(O : X_1, S : X_2, P : X_3, G : X_4, e : X_5, O' : X_6, S' : X_7, P' : X_8), Auth(O : X_1, S : X_2, P : X_9, G : X_{10}, e : X_{11}, O' : X_{12}, S' : X_{13}, P' : X_{14}), role(self : X_2) X_3 \neq X_9$	Les privilège X_3, X_9 ne peuvent pas être affectés simultanément au rôle X_2

TABLE 3.2 – Exemples de règles

3.3.2.3 Avantages

- Le modèle de contrôle d'accès proposé peut être utilisé pour l'analyse et la vérification des autres modèles de contrôle d'accès ;
- Le modèle proposé est plus complète (peut représenter plus de politique) que le modèle [47] ;
- Le formalisme proposé peut exprimer les différentes entités et les règles des modèles de contrôle d'accès existants ;

3.3.2.4 Inconvénients

- Le privilège dans ce modèle de contrôle d'accès ne supporte pas la notion de contrainte ;
- Le formalisme proposé ne peut pas exprimer les politiques de contrôle d'accès qui utilise de la sémantique (ontologie), car la sémantique de la logique des prédicats ne permet pas une telle représentation ;
- Le formalisme la logique des prédicats est compréhensible que par les machines et il est difficile à être interpréter par un humain.

3.3.3 Spécification d'une politique de contrôle d'accès et de certification pour les services web sémantiques

Dans [2], les auteurs présentent comment les autorités de certification peuvent spécifier leurs politiques de certification avec un formalisme lisible par une machine et comment elles peuvent publier la politique ainsi que leur contexte de certification (exemple des propriétés qu'ils ont certifiées), ce formalisme est la logique de description, contrairement

à la logique de prédicat qui est un langage alors que la logique de description est une famille de langages. En effet, actuellement, les autorités de certification présentent leurs politiques de certification explicitement dans des documents qui sont lisibles seulement pour les humains. Ces documents sont destinés à être lus par les fournisseurs de services avant qu'ils définissent les politiques de contrôle d'accès pour leurs services Web.

3.3.3.1 Concept

- **Utilisateur :** représente l'entité qui veut accéder aux services Web. Dans le cas d'un accès restreint, l'utilisateur doit prouver son éligibilité en présentant les informations nécessaires (présenter un certificat par exemple).
- **Fournisseur du service Web :** les fournisseurs du service Web représentent le propriétaire du service Web. Le fournisseur de service Web peut restreindre l'accès à ses services qu'aux utilisateurs autorisés.
- **Autorité de certification :** Les autorités de certification certifient les propriétés de certains utilisateurs en émettant des certificats. Chaque autorité de certification définit sa propre terminologie qu'elle utilise dans ses certificats.

3.3.3.2 Principe du modèle

La figure 3.2 représente les interactions entre les différentes entités du modèle

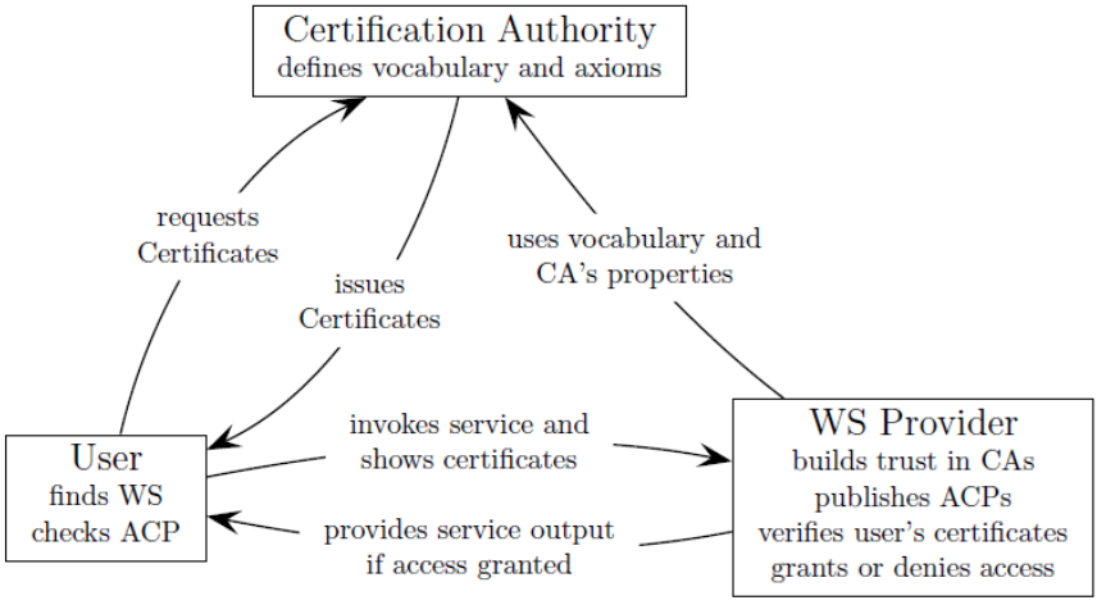

FIGURE 3.2 – Modélisation d'une permission

Un utilisateur qui veut utiliser un service Web, invoque ce dernier en lui montrant son certificat préalablement fourni par une autorité de certification, le fournisseur du service

Web vérifie le certificat de l'utilisateur en utilisant le vocabulaire et les propriétés de l'autorité de certification.

3.3.3.3 Modélisation

Une politique de contrôle d'accès pour un service Web w est un ensemble de termes d'autorisation (p, w, f). Chaque terme d'autorisation a le sens qu'un utilisateur étant en mesure de prouver la propriété p a l'accès à la fonction f du service Web w. Un terme d'autorisation peut être défini avec des axiomes de la logique de description comme suit :

AuthorizationTerm $\sqsubseteq \top \sqcap \exists$subject.ca-Property $\sqcap$
$\exists$ object.WebService $\sqcap$
$\exists$ Authorization.WebServiceFanctionality

Ce qui signifie q'un utilisateur ayant été certifié par la CA a l'autorisation d'accès à la fonctionnalité du service Web.

Exemple : l'exemple suivant illustre la façon dont un fournisseur de service Web permet de spécifier une politique de contrôle d'accès en utilisant les connaissances qu'il acquiert des politiques de certification des autorités de certification. OutShop est un service Web qui offre un service d'enregistrement pour les guides approuvés (trekking). Pour l'enregistrement, ils ont les conditions d'accès suivants : Chaque guide (trekking) doit être approuvé soit par le département des forêts (FD) ou par la faune fondation (WLF), doit être âgé d'au moins de 25 ans et doit être compétent en premier secours. Cela conduit à la politique de contrôle d'accès suivant :

$ACP(P) := \{(WLF, OS.edu, Trecking\ guide), (FD, os.edu, Trecking\ guide)\}$
ou WLF et FS sont comme suit ;
$WLF \equiv WLF.org; \#appguide \sqcap stat.gov; \#above25 \sqcap RedCross.org; \#firstaid$
$FD \equiv FD.org; \#appguide \sqcap stat.gov; \#above25 \sqcap RedCross.org; \#firstaid$

À partir de la politique de certification, le fournisseur de services Web peut déduire que celui qui a le $(FD.org)$ la propriété $(FD.org; \#appguide)$ a également $(state.gov; \#above25)$ et $(RedCross.org; \#firstaid)$. À partir de la spécification de la politique, il lui permet ainsi de réduire le nombre de certificats, un utilisateur potentiel devra présenter :

$ACP(P) := \{(WLF, OS.edu, Trecking\ guide), (FD, os.edu, Trecking\ guide)\}$
avec FS est définie comme
$FD' \equiv FD.org; \#appguide.$

3.3.3.4 Avantages

- Il existe des outils qui permettent aux machines de comprendre les formules générées par la logique de description ;
- La logique de description est convenable pour le Web sémantique.

3.3.3.5 Inconvénients

- La logique de description ne traite que des relations binaires ;
- Les formules logiques générées par la logique de description sont difficiles à interpréter par un humain ;

3.3.4 Formalisme basé sur les graphe pour RBAC

L'idée générale de cet article [51] est de décrire la structure du système de contrôle d'accès RBAC de base [70] à l'aide de la théorie des graphes. Ensuite, considérer les règles et les politiques de contrôle d'accès comme des morphismes afin de spécifier et de vérifier la cohérence des conditions, l'administration des affectations et la révocation des rôles, certaines opérations complexes telles que l'affectation et la révocation des rôles dans une administration décentralisée.

3.3.4.1 Concepts

Les concepts du modèle utilisé sont les concepts du modèle RBAC de base préalablement défini (Rôle, Utilisateur, Permission, Objet, Session) et l'administrateur des rôles qui représente le concept qui gère les différentes opérations effectuées sur les rôles (Affectation des permissions, Affectation des utilisateurs, Révocation d'un rôle... etc.).

3.3.4.2 Modélisation

Afin de modéliser RBAC, les auteurs ont défini un graphe de référence qui représente les différentes entités du modèle et les relations entre ces entités, un graphe valide est un graphe qui représente les différentes entités de ce graphe ainsi que les relations qui les lient. La figure 3.3 illustre le graphe prototype du modèle RBAC :

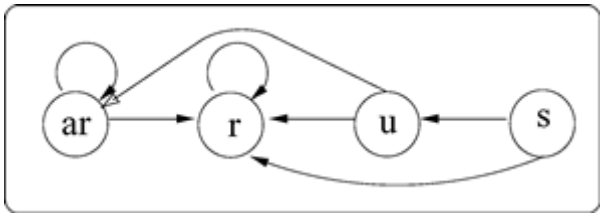

FIGURE 3.3 – Graphe du modèle RBAC

Les concepts sont représentés par des n œuds, les n œuds (ar, r, u, s) représentent respectivement l'administrateur du rôle, le rôle, l'utilisateur, et la session. Un arc entre (s) et (u) signifie que l'utilisateur (u) a une session (s), un arc de l'utilisateur (u) au rôle (r) représente l'affection du rôle r à l'utilisateur (u), un arc de l'utilisateur (u) à l'administrateur (ar) signifie que l'utilisateur (u) est administré par l'administrateur (ar), un arc de l'administrateur (ar) au rôle (r) représente l'administration du rôle (r) par (ar), un arc de la session (s) au rôle (r) représente l'activation de (r) dans (s). la boucle au niveau des n œuds (ar) et (r) est utilisée pour modéliser la hiérarchie de l'administration et la hiérarchie des rôles. Le graphe de la figure 3.3 est utilisé comme prototype pour vérifier la validité d'un graphe d'un modèle RBAC donné.
Un graphe représente l'état d'un système à un moment donné, les intitulés des différentes instances du modèle sont représentés par des annotations sur les n œuds.

Exemple : Dans la figure 3.4, (u1 et u2) représentent des utilisateurs, (s1 et s2) représentent des sessions, (r1, r2 et r3) représentent des rôles et (ar1) représente un administrateur de rôle.

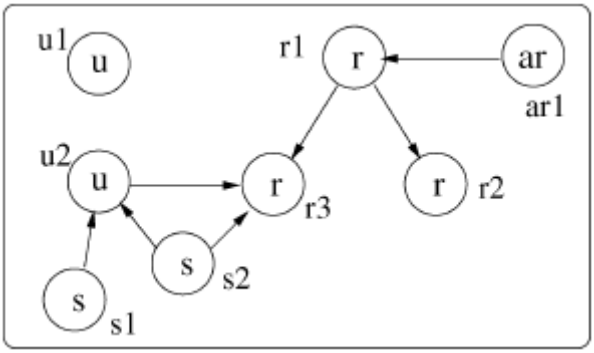

FIGURE 3.4 – Exemple d'un graphe

Les changements d'état sont spécifiés par les règles de transformation de graphe, appelés règles. Une règle formellement est représentée par un morphisme de graphe $r : L \rightarrow R$ où L représente le graphe de départ et R représente le graphe résultant après l'application de la règle. Dans la figure 3.5, l'effet de la règle ajoutée au rôle (add to role) sur le système est montré par la règle de production qui transforme le graphe de gauche vers celui de droite.

La figure 3.6 représente la modélisation des différentes règles applicables sur les rôles dans le modèle RBAC.

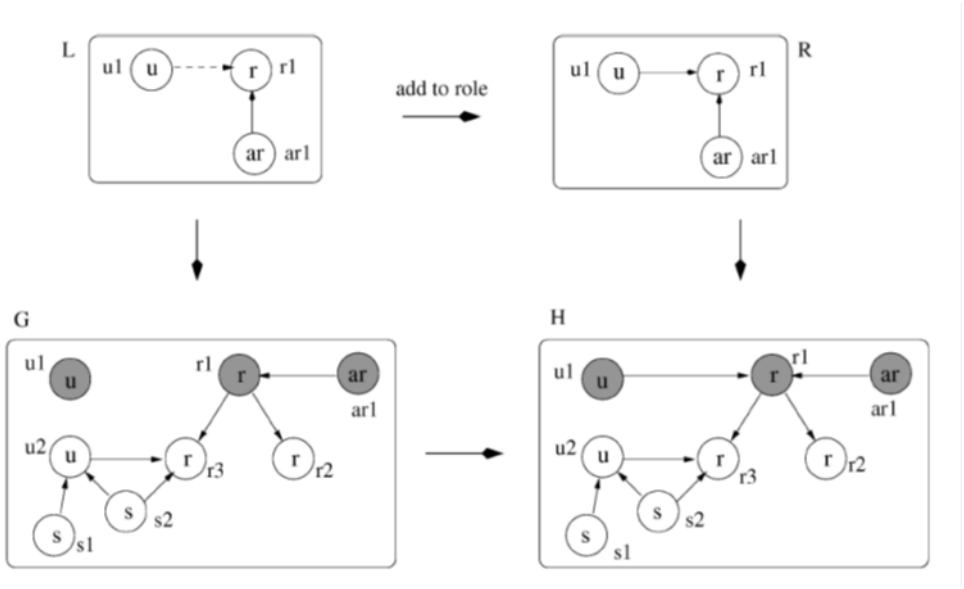

FIGURE 3.5 – Règle ajoutée au rôle et son application

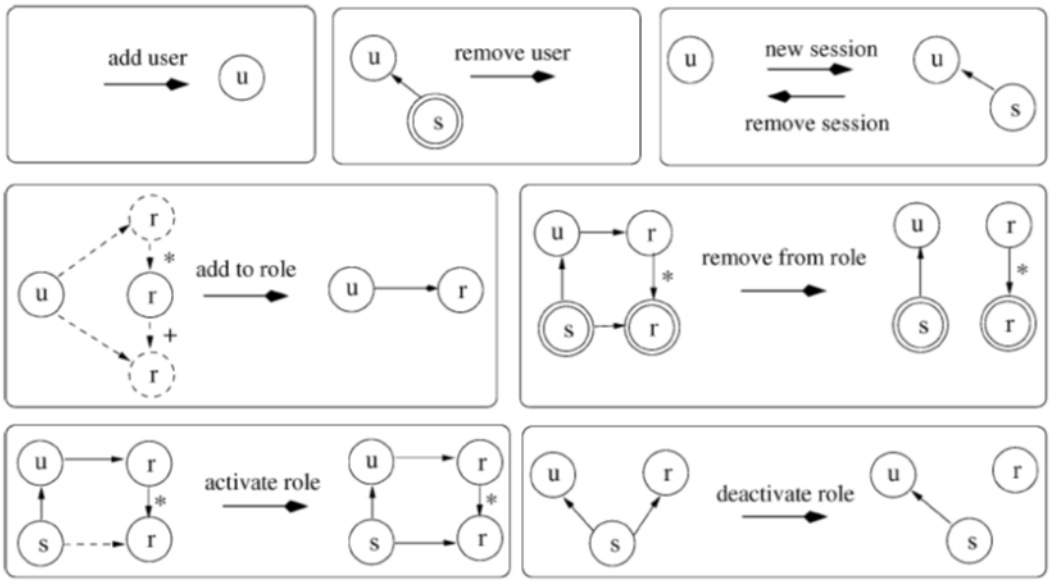

FIGURE 3.6 – Règles applicables au rôle

La différence entre le modèle RBAC décentralisé et le modèle RBAC centralisé est l'administration des rôles, en effet dans RBAC centralisé les rôles sont administrés par un seul administrateur et dans RBAC décentralisé les rôles sont administrés par plusieurs administrateurs organisés sous forme d'une hiérarchie. La figure 3.7 représente une telle hiérarchie.

3.3.4.3 Avantages

- Prédit le comportement du système en le combinant avec différentes politiques de contrôle d'accès ;
- Formaliser avec les graphes donne une description visuelle intuitive des structures

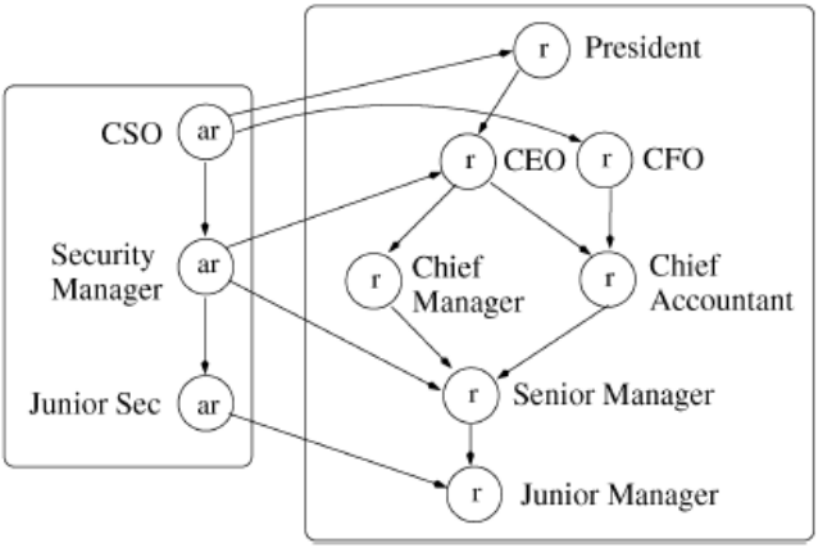

FIGURE 3.7 – Hiérarchie des administrateurs de rôle

dynamiques qui se produisent dans les modèles de contrôle d'accès ;
– L'utilisation des structures de graphique permet un traitement uniforme de l'utilisation des rôles et des administrateurs de rôles [51].

3.3.4.4 Inconvénients

– Formalisé avec les graphes permet de décrire le modèle de contrôle d'accès, mais elle ne fournit pas les outils nécessaires pour le raisonnement sur les différentes étapes de fonctionnement du modèle de contrôle d'accès (comment s'effectue l'attribution des rôles par exemple) ;
– Dans le cas des modèles de contrôle d'accès utilisant de la sémantique telle que les modèles pour les services Web sémantique, l'utilisation des graphes n'est pas convenable, car la sémantique des graphes ne permet pas la représentation des ontologies par exemple et le raisonnement sur ces dernières.

3.3.5 Un cadre pour la vérification de RBAC dans les systèmes à temps réel

Les auteurs [71] proposent une description du modèle RBAC au moyen des réseaux de Petri colorés [48] en considérant certaines contraintes du modèle de contrôle d'accès GTRBAC, sans toutefois intégrer le temps. Le modèle proposé prend en compte les contraintes de cardinalité, de séparation de tâches, de relation d'héritage entre les rôles, de précédence et de dépendance.

3.3.5.1 Modélisation

Le formalisme du modèle RBAC avec les réseaux de Petri colorés est comme suit :

- Les différentes entités du modèle (Rôle, Utilisateurs, Session... etc.) sont modélisées par des couleurs, chaque couleur représente une entité donnée ;
- Les entités et les opérations sont représentées par des jetons (exemple : utilisateur est un jeton $< u >$ de couleur U, l'activation d'un rôle par un utilisateur est représenté par un jeton $< U, R, S >$ de couleur URS ;
- Les informations d'état (Activation d'un rôle, Désactivation d'un rôle, ... etc.) sont modélisées par des places (exemple la place User Role Session activation URS représente l'activation d'une session, un jeton de type $< U, R, S >$ dans cette place signifie que l'utilisateur U a activé la session S avec le rôle R ;
- Les arcs représentent le passage d'un état à un autre.

La figure 3.8 illustre un exemple d'un réseau de Petri coloré du modèle RBAC

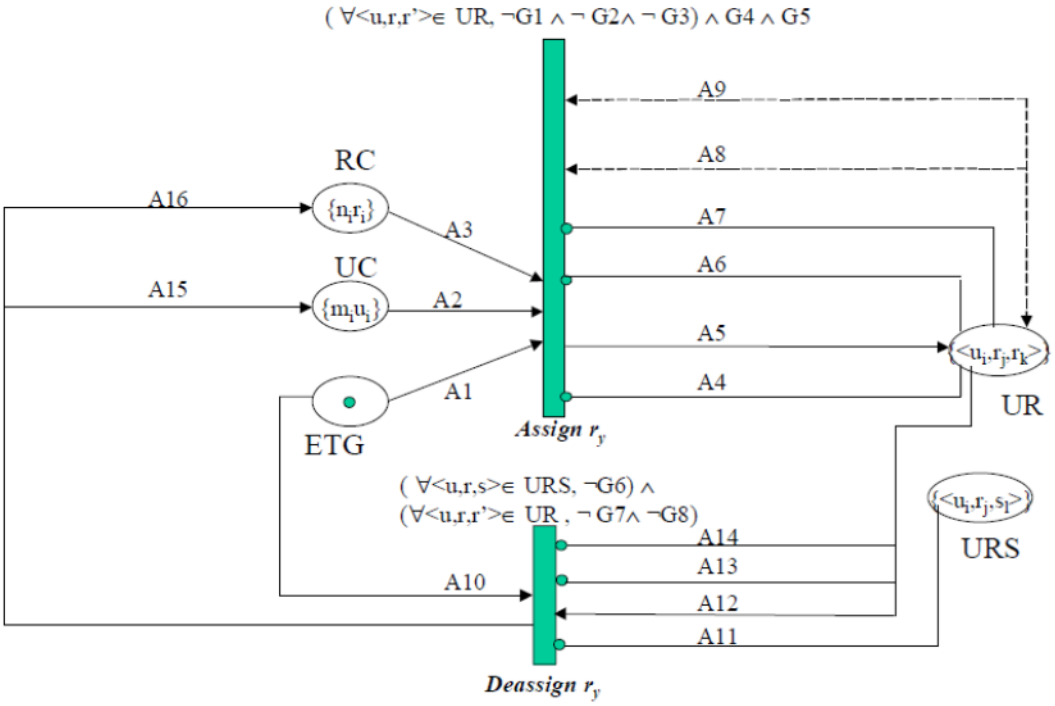

FIGURE 3.8 – RBAC avec les réseaux de Petri colorés

Le but d'une approche basée sur les réseaux de Petri est de prouver que les propriétés ou les règles de la politique de sécurité sont bien renforcées dans le système en bénéficiant des différentes techniques et outils offerts par les réseaux de Petri colorés. La cohérence du modèle RBAC est vérifiée à partir d'une analyse de chaque état accessible du réseau de Petri coloré. L'état est dit cohérent s'il satisfait toutes les contraintes de cardinalité, de séparation de tâches, d'héritage et de précédence et dépendance. Il s'agit ainsi d'effectuer une analyse d'accessibilité qui repose sur le graphe des marquages correspondants au réseau de Petri. En s'assurant de la cohérence du réseau de Petri, on s'assure également

de la garantie des différentes propriétés de sécurité liées à la politique de sécurité.

3.3.5.2 Avantages

- Les réseaux de Petri colorés permettent l'analyse et la vérification d'un modèle de contrôle d'accès à temps réel ;
- Les réseaux de Petri colorés sont adaptés pour modéliser des systèmes de taille importante, cela est dû à l'introduction du concept des couleurs.

3.3.5.3 Inconvénients

- Les réseaux de Petri colorés ne fournissent pas les outils nécessaires pour le raisonnement sur les différentes étapes d'utilisation du modèle de contrôle d'accès ;
- Dans le cas des modèles de contrôle d'accès utilisant la sémantique telle que les modèles pour les services Web sémantique, l'utilisation des réseaux de Petri colorés n'est pas convenable, car la sémantique des réseaux de Petri colorés ne permet pas la représentation des ontologies par exemple et le raisonnement sur ces dernières ;
- La vérification d'un état donné accessible à partir de l'état initial prend un espace et un temps exponentiel.

3.3.6 Modélisation par les automates

Les auteurs [6] proposent un modèle formel basé sur les automates temporisés [3] afin représenter une politique contrat entre deux organisations avec les modalités principales de OrBAC (Permissions, Interdictions, Obligations), les automates temporisés sont pratique pour la synchronisation d'une application distribuée. En effet, la composition d'automates temporisés est obtenue par un produit synchrone : chaque action à exécuter par un automate temporisé correspond à une action avec le même nom à exécuter en parallèle par un autre automate temporisé. En d'autres termes, une transition qui exécute l'action (a) ne peut être déclenchée dans un automate que si une transition étiquetée par l'action (a) peut également être déclenchée dans l'autre automate. Les deux transitions sont déclenchées simultanément, ce qui correspond à un mécanisme de communication par rendez-vous.

3.3.6.1 Modélisation

Dans ce modèle, une permission est simplement représentée par le biais de transitions dans les automates temporisés. Par exemple, dans la figure 3.9 (la figure à gauche), le système peut exécuter l'action (a) à tout moment, peut se comporter selon les possibilités définies dans l'automate A.

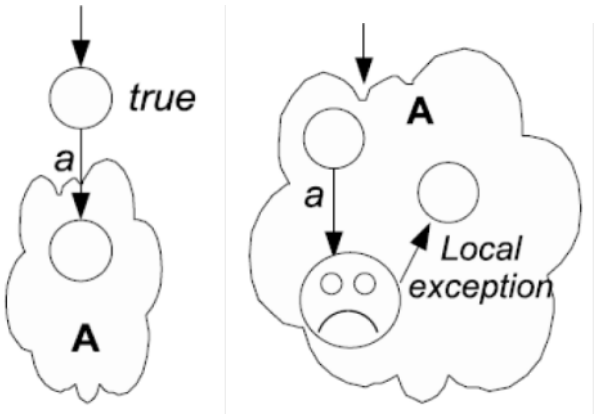

FIGURE 3.9 – Modélisation d'une permission. Modélisation d'une interdiction

Pour les interdictions, deux types d'interdiction ont été utilisées :

- L'interdiction implicite : comme dans les politiques de contrôle d'accès positives, tout ce qui n'est pas explicitement autorisé est interdit, une action qui ne correspond à aucune transition dans l'automate est interdite.
- L'interdiction explicite : dans le modèle, une interdiction explicite est représentée par une transition vers un état d'échec (illustré par une frimousse [3] triste voire la figure 3.9 (figure à droite)).

Les obligations correspondent à des actions qui sont nécessaires dans un certain contexte. D'un point de vue logique, l'obligation est équivalente à une "interdiction de ne pas faire " et le non-respect d'une obligation est donc équivalent à une action interdite. Toutefois, comme les obligations sont sémantiquement plus fortes que les permissions, il faut ajouter d'autres symboles pour décrire cette sémantique et pour faire la distinction entre ce qui est obligatoire et ce qui est simplement autorisé, mais pas obligatoire. Dans la Figure 3.10, (k) est un compteur de temps, (d) est l'échéance temporelle, (b) est l'action obligatoire, (a) est l'action déclenchée par l'échéance, conduisant au traitement d'exception (A2). L'état à partir duquel l'action (b) est obligatoire possède un invariant ($k <= d$).

3. Le terme est approuvé par la Commission générale de terminologie et de néologie, Journal officiel du 16 mars 1998, pour traduire le terme anglais smiley.

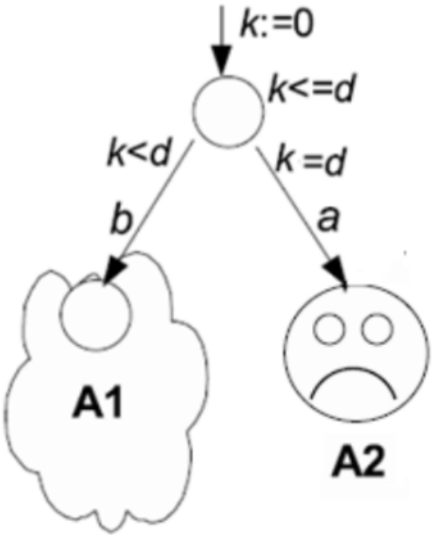

FIGURE 3.10 – Modélisation d'une obligation

La représentation des situations de conflits par les automates temporisés aide à identifier la partie responsable de la violation de politique, par l'analyse de la séquence états transitions qui a conduit à l'état d'échec. L'automate de la figure 3.11 représente une situation de conflit.

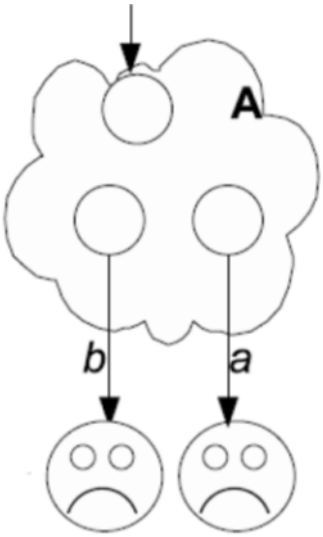

FIGURE 3.11 – Modélisation d'une situation de conflit

3.3.6.2 Avantages

- Les auteurs ont choisi les automates temporisés pour les avantages de simplicité et la richesse suffisante pour exprimer et vérifier les propriétés du modèle.
- Les automates temporisés sont pratique pour la synchronisation ;

3.3.6.3 Inconvénients

- Dans le cas des modèles de contrôle d'accès utilisant la sémantique telle que les modèles pour les services Web sémantique, l'utilisation des graphes n'est pas convenable, car la sémantique des automates temporisés ne permet pas la représentation des ontologies par exemple et le raisonnement sur ces dernières ;
- La taille de l'application augmente exponentiellement vu que le nombre d'états est exponentielle.

3.3.7 Comparaison des modèles de spécification

Dans cette section, nous faisons une comparaison entre les modèles formels pour le contrôle d'accès qu'on a présentés. Avant d'entamer la comparaison, nous présenterons les différents critères.

- **Formalisme :** Représente le formalisme utilisé pour formaliser la politique de contrôle d'accès.
- **Sémantique** : Représente la sémantique de l'outil de modélisation utilisé.
- **Représentation :** Représente la capacité de l'outil de modélisation utilisé à représenter les différentes entités du modèle de contrôle d'accès.
- **Analyse :** Représente la capacité de l'outil de modélisation utilisé à vérifier la politique de contrôle d'accès.
- **Raisonnement :** Représente la capacité de l'outil de modélisation utilisé à raisonner sur les connaissances du modèle formel utilisé.
- **Complexité :** représente la taille engendrée par le formalisme utilisé en raisonnant ou en analysant la politique de contrôle d'accès.

le tableau 3.3 représente la comparaison des différents modèles présentés.

Critères / Modèles	**Jajodia [47]**	**Bertino [14]**	**Agarwal [2]**	**Koch [51]**	**Shafiq [71]**	**Baïna [6]**
Formalisme	Logique des prédicats	Logique des prédicats	Logique de description	Théorie des graphes	Réseaux de Petri Colorés	Les automates temporisés
Sémantique	Non	Non	Oui	Oui	Oui	Oui
Représentation	Suffisante	Suffisante	Suffisante	Suffisante	Suffisante	Suffisante
Analyse	Suffisante	Suffisante	Insuffisante	Insuffisante	Suffisante	Suffisante
Raisonnement	Suffisant	Suffisant	Suffisant	Insuffisant	Insuffisant	Insuffisant
Complexité	moyenne	moyenne	moyenne	Élevé	Élevé	Élevé

TABLE 3.3 – Tableau comparatif des modèles formels pour le contrôle d'accès

3.4 Conclusion

Dans ce chapitre, nous avons présenté différentes approches pour la représentation formelle des modèles de contrôle d'accès. Ces propositions diffèrent principalement dans le genre d'approche (logique ou graphique) utilisée pour représenter les modèles de contrôle d'accès, les politiques et les contraintes. Les approches basées sur les graphes reposent sur l'utilisation des transformations de graphes alors que les approches basées sur la logique se basent sur la programmation logique.

Le prochain chapitre présente la solution que nous avons proposée pour le contrôle d'accès en utilisant les graphes conceptuels. En effet, les graphes conceptuels [82] nous permettront de tirer profit des deux approches. Un graphe conceptuel est un modèle mathématique fondé sur la logique et la théorie des graphes. Cependant, pour raisonner à l'aide des graphes conceptuels deux approches peuvent être distinguées :

- Considérer les graphes conceptuels comme une interface graphique pour la logique et donc raisonner à l'aide de la logique ;
- Considérer les graphes conceptuels comme un modèle de représentation à part entière disposant de ses propres mécanismes de raisonnement fondés sur la théorie des graphes ;

Chapitre 4

Proposition : Spécification d'un modèle formel pour l'expression des autorisations d'accès aux services Web

4.1 Introduction

Contrôler l'accès aux services Web hébergés dans de multiples sources de données distribuées est devenu actuellement un défi pour la communauté industrielle et académique.

A ce jour, aucun standard n'a été proposé pour les modèles de contrôle d'accès pour les services Web. Plusieurs modèles de contrôle d'accès aux services Web proposés tel que WS RBAC [88], X-RABC [19], XGTRBAC [18]... sont des adaptations des modèles de contrôle d'accès aux systèmes d'informations. Toutefois, ces modèles de contrôle d'accès aux services Web dans la littérature supposent que l'invocation de chaque opération fournie par un service Web est indépendante des autres opérations et le contrôle d'accès est appliqué soit au niveau d'une unique opération d'un service Web soit à l'ensemble du service Web [65], et en excluant toute possibilité d'adapter la requête. Dans la pratique, les opérations potentielles qui peuvent être invoquées dépendent de l'état de la conversation en cours entre le client et le service Web, des informations récoltées sur le client et du contexte du client. En plus, le nombre de clients dans la toile contrairement aux systèmes d'information est en constante augmentation.

Dans ce chapitre, nous proposons une spécification d'un modèle formel pour l'expression des autorisations d'accès aux services Web. Dans ce modèle, les services Web sont décrits selon l'ordre d'exécution de leurs opérations (comportemental) [11] [10] pour une granularité plus fine. Afin de réduire la complexité en l'hétérogénéité, les services Web sont regroupés en cercles de confiance selon la politique de contrôles d'accès, où chaque service Web contrôle l'accès à ses propres ressources et afin de masquer l'hétérogénéité

dans les cercles de confiance,nous avons utilisé des ontologies de domaine dans chaque cercle de confiance. Notre politique de contrôle d'accès est basée sur le rôle, l'affectation d'un rôle se fait via le profil du client, ce dernier permet de réduire la fréquence de modification sur l'affectation des rôles. De plus, le modèle permet au client d'adapter ou de compléter sa requête par un processus de négociation.

Afin de modéliser les différents concepts cités et les différentes étapes du processus de contrôle d'accès, nous utilisons un modèle formel qui peut représenter les concepts, exprimer les règles de contrôle d'accès et de résonner sur ces dernières, à savoir les graphes conceptuels [82]. Ces derniers permettent la représentation de connaissances sous forme de graphes et les possibilités de raisonner sur ces connaissances. En effet, les graphes conceptuels sont souvent identifiés comme un langage clé dans la représentation de la connaissance de par leur simplicité, leur richesse formelle, leur expressivité et leur similitude à d'autres langages de modélisation graphiques tels UML [1] [79] ou Entité-Association [29]. Ce formalisme consiste à représenter l'environnement dans lequel on se situe sous forme de concepts, reliés entre eux par des relations. L'intérêt de ces graphes, réside dans le fait qu'ils sont très proches du langage naturel.

4.2 Principe du modèle proposé Access Control for Composed Web Service (AC-CWS)

L'objectif principal du modèle proposé AC-CWS est de fournir un contrôle d'accès dans un environnement distribué qui est l'environnement des services Web. L'objet demandé par le sujet représente un service Web composite, les services Web simples appartenant à ce dernier, collaborent ensemble afin de fournir une permission d'accès au sujet. Les permissions d'accès sont affectées à un rôle, un sujet qui demande une permission d'accès pour une action sur un objet doit activer le rôle qui possède cette permission. L'affectation d'un rôle à un utilisateur se fait selon le profil attribué par le fournisseur du service durant une session. Les permissions d'accès affectées à un rôle dépendent des variables environnementales (Lieu, Temps, etc.), elles peuvent donc changer selon le contexte utilisateur.

La figure 4.1 illustre les différents concepts du modèle proposé AC-CWS.

1. Unified Modeling Language

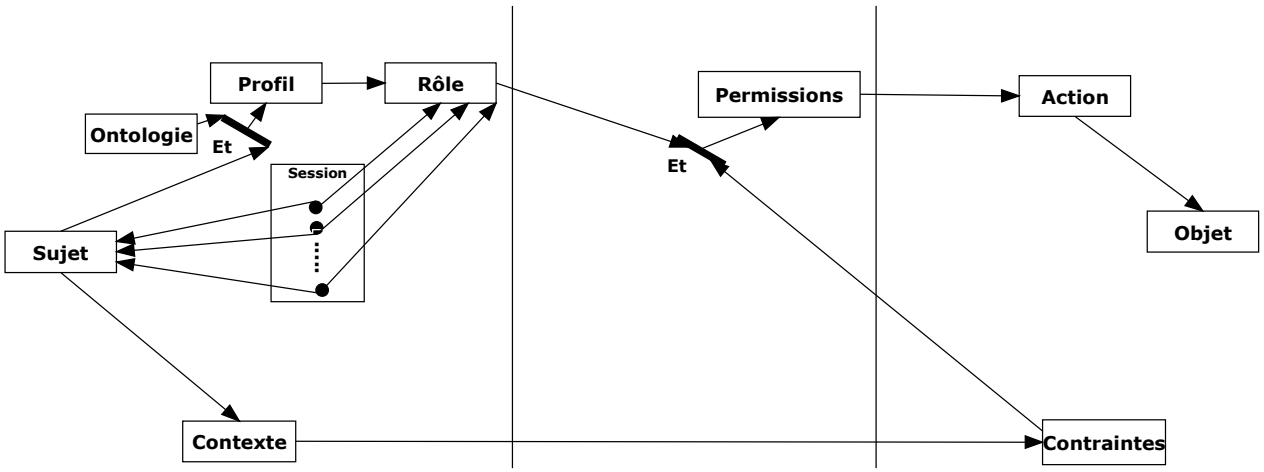

FIGURE 4.1 – Politique de AC-CWS

4.3 Architecture du modèle de contrôle d'accès AC-CWS

Nous représentons l'architecture du modèle proposé par la figure 4.2. Les différents éléments intervenants sont :

- **Sujet :** est le demandeur d'accès à un service Web composite ;
- **Requête composite** : est constituée d'un ensemble de requêtes simples vers les services Web simples correspondants ;
- **Modèle de contrôle d'accès :** est l'élément qui va vérifier la permission d'accès à la requête composite.
- **Objet :** est un service Web composite comportemental ;

4.4 Description des éléments de AC-CWS

4.4.1 Description du modèle

Le modèle de contrôle d'accès proposé AC-CWS est un modèle de contrôle d'accès aux services Web composites décrits sémantiquement. La vérification d'une requête implique la participation des différents services Web simples qui participent à la satisfaction de la requête. Dans le modèle proposé, chaque politique de contrôle d'accès est définie par le fournisseur du service Web simple. En effet, ceci est dû à de nombreux facteurs, certaines politiques par exemple seront plutôt axées sur le niveau de confidentialité des données comme pour le secteur militaire ou dans le domaine commerciale où on privilégie en premier l'intégrité des ressources. Cela engendre plusieurs politiques hétérogènes. Afin

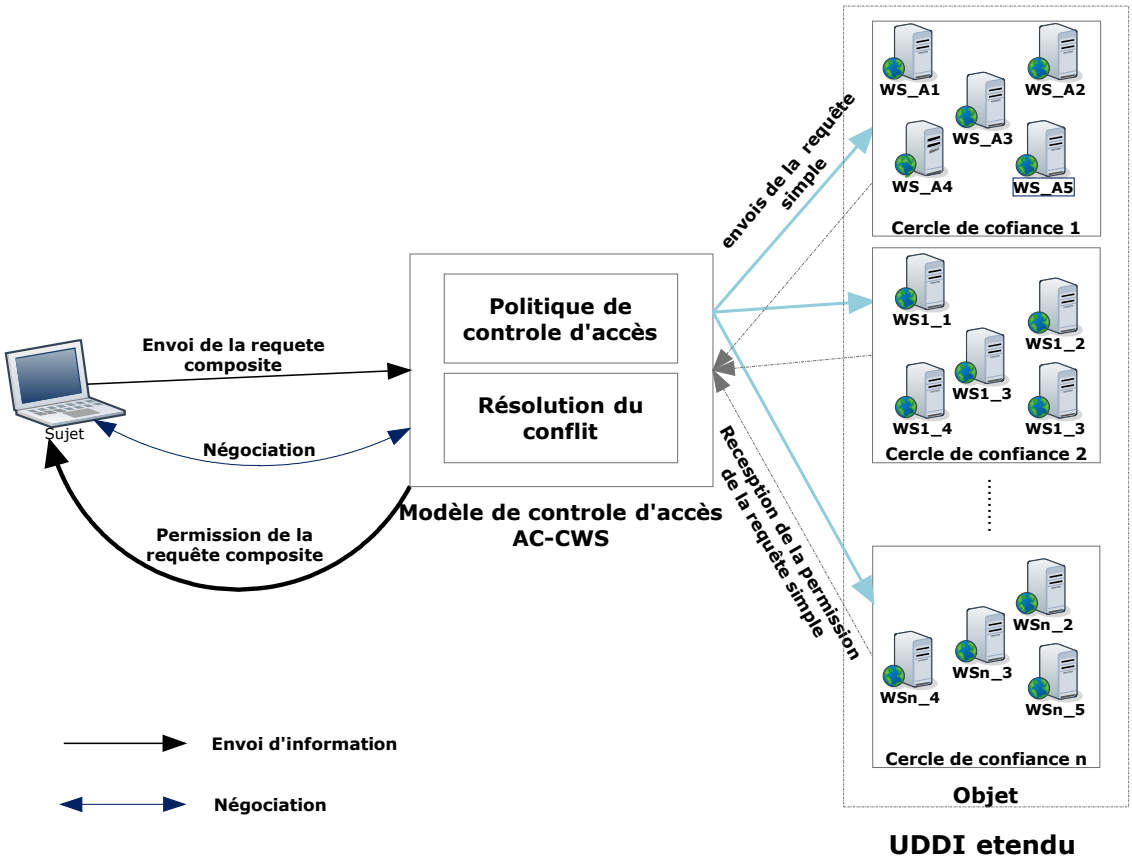

FIGURE 4.2 – Architecture de AC-CWS

de résoudre le problème d'hétérogénéité, les services Web simples sont regroupés selon la politique de contrôle d'accès utilisée sous forme d'un cercle de confiance. La requête du sujet est destinée à un service Web composite, cette dernière est décomposée en sous-requêtes (requêtes simples) destinées aux différents services Web simples impliqués dans la satisfaction de la requête composite. Les permissions d'accès dans le modèle proposé sont affectées à des rôles prédéfinis par le fournisseur du service Web simple dont l'attribution des rôles se fait via le profil du sujet. Ainsi pour avoir la permission d'accès, le profil du sujet doit appartenir à un rôle qui contient cette permission et qui satisfait les contraintes imposées. Les permissions d'accès issues des différents services Web simples sont ensuite adaptées à notre politique de contrôle d'accès par une ontologie de domaine pour enfin transmettre la permission de la requête composite de base. Dans certains cas, le modèle proposé propose au sujet de compléter sa requête via une négociation.

Le modèle AC-CWS définit les éléments suivants :

- S, Pr, R, C, P, O, Se, Rq représentent respectivement l'ensemble de sujets, de profils, de rôles, de contextes, de permissions, de objets, de sessions et de requêtes ;
- $SA \subseteq S$ X P : est une relation associant un sujet à un profil ;

- $RA \subseteq Pr$ X R : est une relation associant à chaque profil un rôle ;
- $PA \subseteq R$ X P : est une relation plusieurs à plusieurs associant les permissions aux rôles ;
- $CA \subseteq S$ X C : est une relation plusieurs à plusieurs associant les contextes aux sujets ;
- $Session - Sujet : Se \rightarrow R$ est une fonction associant chaque session (se_i) à un sujet ($Sujet(se_i)$) ;
- $Session - Role : Se \rightarrow R$ est une relation associant chaque session (se_i) à une rôle ($Role(se_i)$).

Dans ce qui suit, une description détaillée des différents éléments de AC-CWS sera donnée.

4.4.2 Sujet

Le sujet ($s \in S$) est le demandeur d'accès à un service Web composite ($o \in O$), il peut être un humain, un agent logiciel, un service Web, ... etc. Les informations du sujet sont représentées sous forme d'un ensemble de couples (Attribut, Valeur) où :

- Attribut représente une information personnelle du sujet tel que (nom, prénom, profession... etc.) ;
- Valeur représente la valeur de l'attribut.

Un sujet (s) est représenté comme suit : $s = \{(A_1, V_1), (A_2, V_2), ...(A_n, V_n)\}$ où A_i et V_i, $i = 1, .., n$ représente respectivement le *ieme* nom de l'attribut et la *ieme* valeur de l'attribut du sujet (s) et (A_i, V_i) représente le couple attribut valeur.

Exemple : Le couple (age : 20) représente l'âge du sujet.

Le sujet représente l'identité du demandeur de la requête d'accès. Ces informations ne suffisent pas à satisfaire les contraintes d'accès, car il faut ajouter des informations supplémentaires telles que le contexte utilisateur et l'historique du sujet. Au niveau d'un service Web, les sujets se présentent sous forme de profils et pour avoir une permission d'accès, le sujet doit avoir un rôle qui contient cette permission. Dans ce qui suit, nous définirons les différents éléments relatifs au sujet.

4.4.2.1 Contexte utilisateur

Le contexte utilisateur ($c \in C$) représente l'ensemble des caractéristiques de l'environnement physique ou virtuel du sujet (s) qui affecte le comportement d'une application et dont la représentation et l'acquisition sont essentielles à l'adaptation des informations et des services [67]

Le contexte utilisateur dans le modèle proposé AC-CWS est utilisé comme une condition de satisfaction d'une permission d'accès donnée, car le contrôle d'accès n'est pas statique. En effet, les permissions d'accès dépendent des conditions qui, si elles sont satisfaites, permettent d'activer dynamiquement ses permissions. Dans ce cas, on parle souvent de permissions contextuelles. Ainsi, les permissions peuvent dépendre du contexte temporel (par exemple permission pendant les heures de travail), contexte géographique (par exemple, permission à l'intérieur de l'enceinte sécurisée de l'entreprise). D'autres types de contextes peuvent également être définis selon les contraintes imposées par le fournisseur de service.

Le contexte est représenté par un ensemble de couples (Attribut, valeur) dont :

- Attribut représente le nom du contexte utilisateur ;
- Valeur représente la valeur du contexte utilisateur.

Un contexte utilisateur (c) est représenté comme suit : $c = \{(A_1, V_1), (A_2, V_2), ...(A_n, V_n)\}$ où A_i et V_i, $i = 1, .., n$ représente respectivement le *ieme* nom du contexte utilisateur et la *ieme* valeur du contexte utilisateur du sujet s et (A_i, V_i) représente le couple attribut valeur.

Exemple : L'exemple suivant représente le contexte utilisateur ($c1$), qui est composé de deux contextes ($Lieu$) et ($Saison$).
$c1 = \{$(Lieu, Béjaia), (Saison, Hiver)$\}$.

4.4.2.2 Profil

Un sujet (s) dans le modèle proposé AC-CWS est identifié par un profil ($pr \in Pr$). Un profil (pr) est prédéfini dans le système et est représenté par 3 dimensions distinctes [89] :

1. **Attributs statiques :** ce sont les données personnelles, elles comprennent l'identité du sujet (nom, prénom, numéro de sécurité sociale... etc.), des données démographiques (date de naissance, genre... etc.). Les données personnelles sont stables dans le temps. Les attributs statiques sont représentés par un ensemble de couples (Attribut, Valeur) où l'attribut représente l'intitulé de la donnée et la valeur représente la valeur de la donnée ;
2. **Attributs dynamiques :** ce sont les données qui sont variables dans le temps, elles comprennent la situation professionnelle (profession, grade, échelant... etc.), les données géographiques (lieu actuel, heure locale, etc.), etc. Les données dynamiques sont représentées par un ensemble de couples (Attribut, Valeur) où l'attribut représente l'intitulé de la donnée et la valeur représente la valeur de la donnée ;

3. **Attributs de comportement :** ce sont les données sur le sujet des sessions précédentes récoltées et conservées par le service Web (Historique de l'utilisateur). Les attributs du comportement sont représentés par un ensemble de couples (Attribut, Valeur), dont l'Attribut représente le nom de la donnée et la valeur représente la Valeur de l'attribut.

Le profil sert d'intermédiaire entre le sujet et le rôle. L'utilisation du profil comme intermédiaire réduit la fréquence de modification des affectations rôle-utilisateur engendrées par les changements constants des données du sujet. Et il réduit la charge des sujets sur le rôle, vu que les services Web doivent gérer une quantité importante d'utilisateurs.

Un profil (p) est représenté par $pr = \langle Statique, Dynamique, Historique \rangle$ où :

- $Statique = \{(As_1, Vs_1), (As_2, Vs_2), ...(As_n, Vs_n)\}$ tel que As_i et Vs_i , $i = 1, .., n$ représente respectivement le *ieme* nom d'attribut statique et la valeur du *ieme* nom d'attribut statique du profil pr et (As_i, Vs_i) représente le couple attribut valeur.
- $Dynamique = \{(Ad_1, Vd_1), (Ad_2, Vd_2), ...(Ad_n, Vd_n)\}$ tel que Ad_i et Vd_i , $i = 1, .., n$ représente respectivement le *ieme* nom d'attribut dynamique et la valeur du *ieme* nom d'attribut dynamique du profil pr et (As_i, Vs_i) représente le couple attribut valeur.
- $Historique = \{(Ah_1, Vh_1), (Ah_2, Vh_2), ...(Ah_n, Vh_n)\}$ tel que Ah_i et Vh_i , $i = 1, .., n$ représente respectivement le nom de la *ieme* donnée sur le sujet (s) et la valeur de Vh_i et (Ah_i, Vh_i) représente le couple attribut valeur.

4.4.2.3 Rôle

Le rôle est l'entité centrale d'AC-CWS, un rôle ($r \in R$) est une collection particulière de profils ($prs \subseteq Pr$) et de permissions ($ps \subseteq P$) sont réunis par un rôle (r). Le rôle (r) est plus stable vu que les activités et les opérations d'un service Web ne changent pas souvent.

Les permissions d'accès sont affectées au sujet (s) par l'intermédiaire d'un rôle (r). Le rôle (r) représente le sujet (s) à une session donnée dont l'attribution d'un rôle (r) à un sujet (s) dépend du profil(pr). Afin de réduire le nombre de permissions affectées à un rôle (r), les rôle sont organisés en hiérarchie [70], ainsi les rôle hérite des permission des rôles hiérarchiquement inférieures.

Un rôle (r) est représenté par $r = \langle prs, ps, rs \rangle$, $prs \subseteq Pr$ représente l'ensemble des profils affectés à ce rôle, $ps \subseteq P$ représente l'ensemble des permissions affectées à ce rôle et $rs \subseteq R$ représente l'ensemble des rôles hérités.

4.4.2.4 Session

Lorsque le sujet émet une requête d'accès, une session ($se_i \in Se$) lui sera affectée, durant toute cette session, le sujet peut utiliser les permissions d'accès affectées via un rôle. Une fois la session (se_i) expirée, les permissions d'accès affectées au sujet via le rôle ne seront plus valables.

4.4.3 Objet

L'objet ($o \in O$) est une entité passive du système, c'est sur ce dernier que s'effectuent les tâches demandées par le sujet (s). L'objet est un service Web composite qui est constitué d'un ensemble de services Web simples qui sont regroupés en cercles de confiances dans le répertoire universel UDDI.

4.4.3.1 Service Web simple

Un service Web simple (sw) représente l'entité qui va satisfaire une partie de la requête ($rq \in Rq$) du sujet (s), il contient un politique ($pl \in Pl$) de contrôle d'accès qui gère les permissions d'accès à ses ressources. Les services Web simples (sw) se basent sur le modèle comportemental, c'est-à-dire qu'ils sont représentés par leurs opérations et leurs paramètres.
Un service Web simple ($sw \in SW$), où (SW) représente l'ensemble des services Web, est représenté par ($sw = \langle Prm, Opr \rangle$) où ($Prm$) et ($Opr$) représente respectivement l'ensemble des paramètres et l'ensemble des opérations du service Web simple (sw). Une opération ($opr \in Opr$) est représentée par ($opr = (In, Out, pl)$) tel que (In) et (Out) représente respectivement l'ensemble des paramètres en entré et l'ensemble des paramètres en sortie. Chaque paramètre ($prm \in Prm$) ou opération ($Opr \in opr$) est doté d'un niveau de sensibilité (sensible ou non sensible). La sensibilité intervient dans l'héritage des permissions, car les permissions d'un paramètre ou une opération sensible ne sont pas héritables, et ($pl \in Pl$) représente la politique de contrôle d'accès du service Web (sw).

4.4.3.2 Service Web composite

Un service Web composite représente l'objet qui va satisfaire l'ensemble de la requête du sujet, il représente un service Web virtuel qui est composé d'un ensemble de services Web simples selon l'ordre de leurs opérations et qui interagissent entre eux afin de satisfaire la requête du sujet. Le service Web composite spécifie quels services ont besoin d'être invoqués, dans quel ordre et comment gérer les conditions d'exception.

Un service Web composite ($o \in O$) est représenté par ($o = (o_1, o_2, ..., o_n)$) où ($o_i = sw.op$, $i = 1, .., n$) tel que ($sw \in SW$) représente un service Web qui contribue à la

satisfaction de la requête (rq) et ($op \in (Prm \cup Opr)$) représente une opération ou un paramètre du service Web sw.

4.4.3.3 UDDI étendu

L'UDDI définit le mécanisme permettant de répertorier les services Web simples, et de gérer les informations de publication, de découverte et l'utilisation des services. Les services Web au niveau de l'UDDI étendu sont regroupés en cercle de confiance. Un cercle de confiance représente un ensemble de services Web simples qui utilise la même politique de contrôle d'accès, ceci permet de réduire la complexité d'hétérogénéité. En effet, chaque politique de contrôle d'accès utilise sa propre syntaxe et sa propre sémantique.

L'annuaire universel de description, de découverte et d'intégration étendu est représenté par un ensemble de cercles de confiances comme suit : $Uddi = \{ccf_1, ccf_2, ..., ccf_n\}$ où (ccf_i, $i = 1, .., n$) représente un cercle de confiance tel que ($ccf_i = (sw_1, sw_2, ..., sw_m)$)où ($sw_j, j = 1, .., m$) représente un service Web et les politiques de contrôle d'accès sont équivalentes ($ws_1.pl \equiv ws_2.pl \equiv ... \equiv ws_m.pl$).

4.4.4 Requête

Une demande d'accès est une requête ($rq \in Rq$) qui spécifie les éléments suivants :
- L'identité du demandeur qui est un sujet ($s \in S$) ;
- La ressource à accéder Objet ($o \in O$) ;
- L'action ($a \in A$).

Dans le modèle proposé AC-CWS la requête du sujet est représentée comme suit :
($rq = (s, [(o_i, a_i)Opt]*)$) tel que :

1. **(s) :** est le sujet ou demandeur de la requête ;
2. **(o_i) :** représente un élément de l'objet o préalablement défini ;
3. **(a) :** est l'action à effectuer sur (o_i) tel que ($a_i \in \{read, write, execute\}$) ;
4. **(Opt) :** représente l'option de composition, les différentes options de composition sont :
   - En série noté « - » : L'exécution des actions se fait en série ;
   - Au choix noté « + » : L'exécution des actions se fait au choix ;
   - Nul : est utilisé seulement à la fin.
5. **((o_i, a_i)) :** représente une requête simple destinée à un seul service Web, la requête simple rq_i est sous la forme ($(s, (o_i, a_i))$) ;
6. ***** : représente l'itération des actions demandées, en effet une requête peut contenir plusieurs requêtes simples ;

7. **([$(o_i, a_i)Opt$]$*$) :** représente la requête (Composite) destinée à un service Web composite ;

Exemple : [(WS_Banque. Consulté, lire) - (WS_Banque. Retiré, exécuté)].

4.4.5 Règle de contrôle d'accès

Nous pouvons définir une règle de contrôle d'accès comme étant un ensemble de spécifications qui répondent aux questions suivantes :

- Quels sont les sujets concernés ?
- Quels sont les objets accédés ?
- Quelles actions demandées ?
- Y a-t-il d'autres contraintes à satisfaire ?
- Quelle permissions renvoyer ?

Une règle de contrôle d'accès définit un ensemble de contraintes et une permission. La permission peut être soit positive, pour permettre l'accès à la ressource (permit), soit négative, pour en refuser l'accès (deny) soit partiellement satisfaite (p-permit). La permission constituent la réponse d'une règle sous la forme ($p \in P$) dans le cas où les contraintes ($c \in C$) imposées sont satisfaites.
Le système de contrôle d'accès proposé doit évaluer les données récoltées sur le sujet et générer une permission. La manière la plus directe d'aboutir à une permission est de vérifier si les attributs des sujets, objets et actions correspondent à des valeurs particulières. Nous parlons dans ce cas de cible (target), qui représente une première étape pour savoir si une règle peut être appliquée à une requête ou non.
Une règle peut imposer un ensemble de contraintes supplémentaires comme le contexte. Ainsi, si une requête correspond à la cible d'une règle, alors les conditions de cette règle seront évaluées, et si elles sont satisfaites, la réponse sera l'effet spécifié. Sinon, si la cible d'une règle ne correspond pas à la requête, ou bien si ses conditions ne sont pas satisfaites alors cette règle ne sera pas appliquée.

Dans le modèle AC-CSW, une règle de contrôle d'accès (rg) est sous la forme suivante : ($si\ condition\ alors\ permission$), cela signifie que pour avoir la permission, il faut satisfaire la condition, dont ($condition = (r, cs)$) où ($r \in R$) est le rôle attribué au sujet(s , $cs \subseteq C$) représente le contexte de l'utilisateur (contrainte à satisfaire) tel que ($cs = (c_1, c_2, ..., c_n)$ où ($c_i \in C$, $i = 1, .., n$) représente un contexte utilisateur. ($Permission$) représente la permission d'accès tel que permission est représenté par ($permission = (p, a, o_i)$) dont ($p \in P$) représente la permission d'accès attribuée et ($a \in A$) représente l'action à effectuer et ($o_i \in o$) représente l'opération ou le paramètre du service Web demandé par le sujet (s).

4.4.6 Politique de contrôle d'accès

Une politique de contrôle d'accès définit les permissions sur les opérations et les paramètres d'un service Web simples ainsi que les rôles habilités à utiliser ces permissions. Un politique de contrôle d'accès représente l'ensemble des règles appliquées afin d'assurer la sécurité du service Web simple.
Dans le modèle proposé, une politique de contrôle d'accès ($p \in P$) (tel que (P) représente l'ensemble des politiques du modèle) est représente par ($p = (R, O, A, Rg)$) tel que (R) représente l'ensemble des Rôles, (O) représente l'ensemble des objets, (A) représente l'ensemble des actions et (Rg) représente l'ensemble des règles applicable par les Rôle de l'ensemble (R) sur les objets de l'ensemble (O) avec les actions de l'ensemble (A).

Exemple : Soit la politique : $p_1 = (\{Directeur, Agent, Client\}, SW\_Banque, \{Read, Write, Execut\}, \{rg_1, rg_2, rg_3\})$. Dans cet exemple la politique contient trois rôle ($Directeur, Agent, Client$), l'objet représente le service Web (SW_Banque), les actions sont ($Read, Write, Execut$) et les règles applicables sont (rg_1, rg_2, rg_3).

4.5 Graphe conceptuel pour AC-CWS

Dans cette section nous présentons la modélisation des entités d'AC-CWS par les graphes conceptuels. Cette représentation sera utilisée pour réaliser les différentes étapes du processus de contrôle d'accès d'AC-CWS. Avant de présenter la modélisation des différentes entités, nous présentons les notions de base des graphes conceptuels que nous utiliserons.

4.5.1 Notions de base des graphes conceptuels

Les graphes conceptuels sont un modèle de représentation de connaissances du type réseau sémantiques qui a donné lieu à un certain nombre de travaux depuis son introduction par John F. Sowa en 1984 [82].

4.5.1.1 Graphe conceptuel

Un graphe conceptuel est un graphe étiqueté, biparti, connexe. Il est composé : (i) d'un ensemble de sommets concepts (notés par des rectangles) qui représentent les entités, attributs, événements, (ii) d'un ensemble de sommets relations (notés par des ovales) qui expriment la nature des liens entre les concepts, (iii) d'un ensemble d'arcs qui lient les sommets relations aux sommets concepts, (iv) d'une étiquette associée à chaque sommet (un couple (type de concept, marqueur) pour un sommet concept, un type de relation pour un sommet relation, le numéro d'ordre dans le voisinage d'un sommet relation donnée pour un arc).

4.5.1.2 Support

Les relations et les concepts sont ordonnés dans une structure de treillis orienté du plus spécifique au plus général appelé support. Le support contient le vocabulaire utilisé pour construire la base de connaissances : les types de concepts utilisés, les instances de ces types, et les types de relations permettant de relier les concepts. L'ensemble des types de concepts TC est partiellement ordonné par la relation (sorte de). Universel et Absurde en sont respectivement le plus grand et le plus petit élément [83].
Les concepts peuvent être liés par des relations. Le support contient l'ensemble des types de relations TR, qui est partiellement ordonné par la relation (sorte de).

4.5.1.3 Individu

L'ensemble des marqueurs individuels contient quant à lui les instances de concepts. Le marqueur générique (noté *) est un marqueur particulier référençant une instance de concept indéfinie .

4.5.1.4 Headed graph

Un graphe conceptuel est de type headed graph s'il dispose d'un certain n œud choisi comme une tête sémantique [63].

4.5.1.5 Représentation des graphes conceptuels

Plusieurs représentations des graphes conceptuels sont possibles, nous citons les trois notations les plus connues : la représentation linéaire (Linear Form), la notation CGIF [2], la notation graphique (Display Form) que nous utiliserons dans notre proposition.

La notation graphique (Display Form) : Un graphe conceptuel représente une formule logique les noms et les arguments sont représentés par des n œuds. Les arcs du graphe relient les noms des prédicats à leurs arguments. On utilise des rectangles pour les concepts (arguments) et des cercles pour les relations (prédicats). Cette représentation est dite représentation graphique d'un graphe conceptuel.

Exemple : La figure 4.3 illustre un graphe conceptuel avec la notation graphique, dans cet exemple, le rôle représente la tête sémantique du graphe.

4.5.1.6 Les règles de graphes

Les règles des graphes sont introduites pour la première fois par E.Salvat [68] comme une extension des graphes conceptuels, proposés pour exprimer des connaissances de type

2. Conceptual Graph Interchange Format

FIGURE 4.3 – Exemple d'un graphe conceptuel

"*Si* $G1$ *alors* $G2$" ou "$G1 \rightarrow G2$" tel que "$G1$" et "$G2$" sont des graphes conceptuels. Formellement, une règle de graphes est constituée d'un graphe hypothèse "$G1$", d'un graphe conclusion "$G2$", et d'un ensemble de points d'attache correspondant à des liens de connexion entre "$G1$" et "$G2$".

4.5.2 Représentation des différentes entités du modèle proposé par les graphes conceptuels

4.5.2.1 Représentation du sujet

Le sujet est sous forme d'un graphe conceptuel de type headed graph, la tête sémantique est représentée par un concept sujet, les couples (Attribut, Valeur) sont représentés respectivement par les concepts de type (Attribut) et (Valeur). La relation (est constitué) qui relie le sujet à l'attribut signifie l'appartenance d'un attribut à un sujet, et la relation (À une) indique la valeur d'un attribut.

Exemple : La figure 4.4 représente un sujet dont les attributs sont : nom, âge et niveau et leurs valeurs sont respectivement (Hakim, 25 et bac + 5).

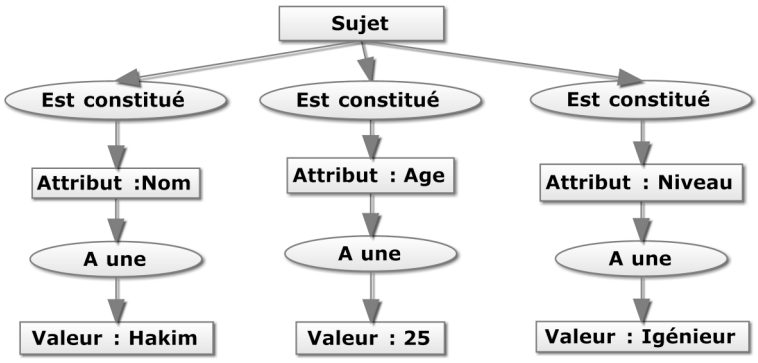

FIGURE 4.4 – Exemple d'un graphe conceptuel représentant un sujet

4.5.2.2 Représentation du profil

Le profil est représenté par un graphe conceptuel de type headed graphe où le concept (Profil) représente la tête sémantique. Les dimensions du profil sont représentées par les concepts de types (Statique, Dynamique et Historique). Ces concepts représentent respectivement l'ensemble des attributs statiques et dynamiques du sujet et les données supplémentaires du service Web sur l'utilisateur. Les couples (Attribut, Valeur) sont représentés respectivement par les concepts de type (Attribut) et (Valeur) reliés par la relation (a une) qui signifie l'attribut a une valeur, les couples (Attribut, Valeur) sont représentés respectivement par les concepts de type (Attribut) et (Valeur) reliés par la relation (a une) qui signifie l'Attribut a une valeur. La relation (est constitué) reliant le concept statique avec le concept attribut signifie que l'attribut appartient à la dimension statique, de même pour les relations (est constitué1) qui signifient l'appartenance d'un attribut à la dimension dynamique. Pour la relation (est constitué2), elle relie le concept historique avec le concept Attribut, qui signifie l'appartenance de l'Attribut a l'historique.

Exemple : La figure 4.5 représente un profil intitulé (abonné), la dimension statique de ce profil est constituée par les attributs (nom et âge), la dimension dynamique est constituée par l'attribut (niveau), et la dimension historique est constituée par l'attribut (nombre de connexion).

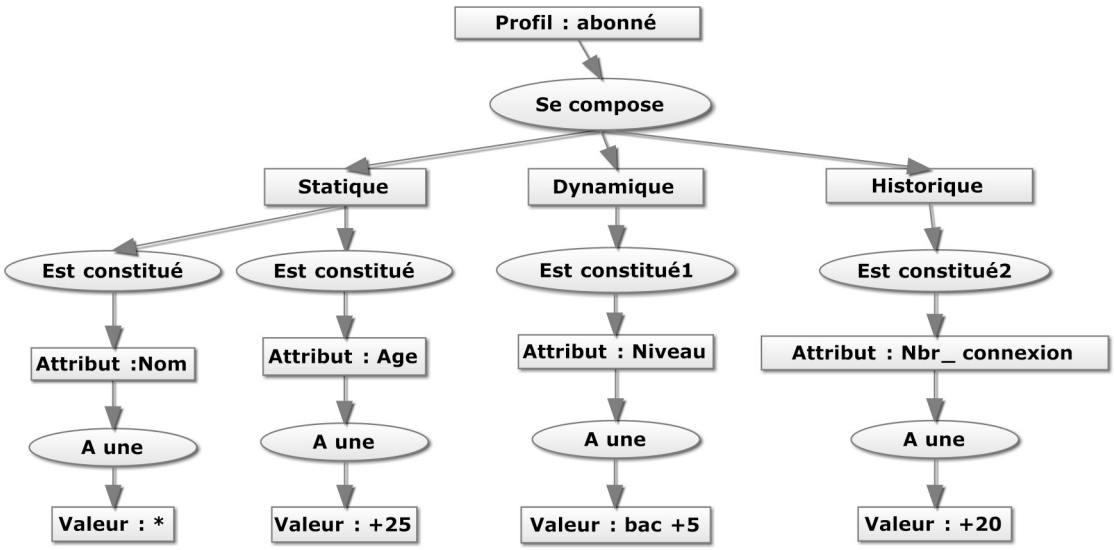

FIGURE 4.5 – Exemple d'un graphe conceptuel représentant un profil

4.5.2.3 Représentation du rôle

La hiérarchie et les affectations profil-rôles sont représentées par un graphe conceptuel, le rôle est représenté par un concept intitulé rôle, la relation d'héritage est représentée par la relation (hérite) et l'affection d'un profil-rôle est représentée par la relation (contient).

Exemple : la figure 4.6 représente un exemple d'une hiérarchie de rôle avec les profils qui sont affectés à ces rôles.

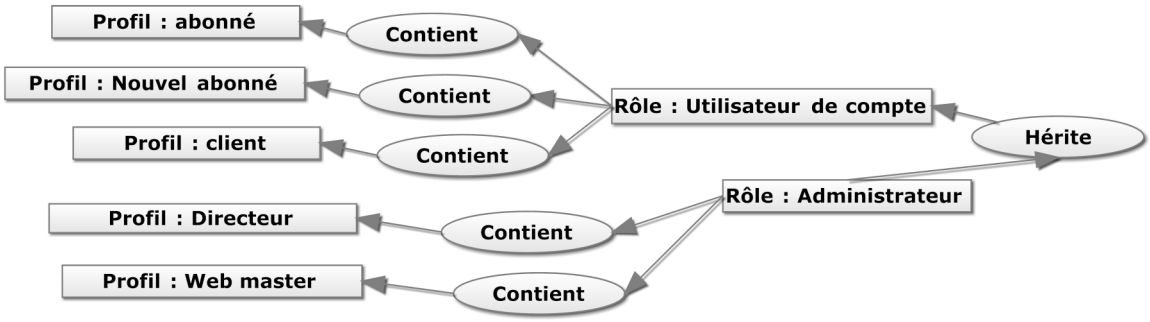

FIGURE 4.6 – Exemple d'un graphe conceptuel représentant un rôle

4.5.2.4 Représentation du contexte utilisateur

Le contexte utilisateur est un graphe conceptuel de type headed graphe dont le concept (Contexte utilisateur) représente la tête sémantique, les couples (Attribut, Valeur) sont représentés respectivement par les concepts de type (Attribut) et (Valeur) reliés par la relation (a une) qui signifie l'attribut a une valeur.

Exemple : la figure 4.7 illustre un exemple de contexte utilisateur qui est composé d'un contexte (saison) et d'un contexte (lieu) avec les valeurs respectives (Hiver) et (Bejaia).

4.5.2.5 Représentation des services Web simples

Le service Web simple est représenté par un graphe conceptuel de type headed graph dont le concept (service Web) représente la tête sémantique, les opérations et les paramètres sont représentés respectivement par les concepts de type (Opération) et (Paramètre). Les entrées et les sorties des opérations sont représentées respectivement par les concepts (Entrée) et (Sortie). La relation (Composer) lie le concept (Service Web) et le concept (Opération) qui signifie que l'opération appartient au service Web, de même pour la relation (Composer1) avec le concept (Opération). La sensibilité des opérations et des paramètres est représentée par le concept (Sensible).

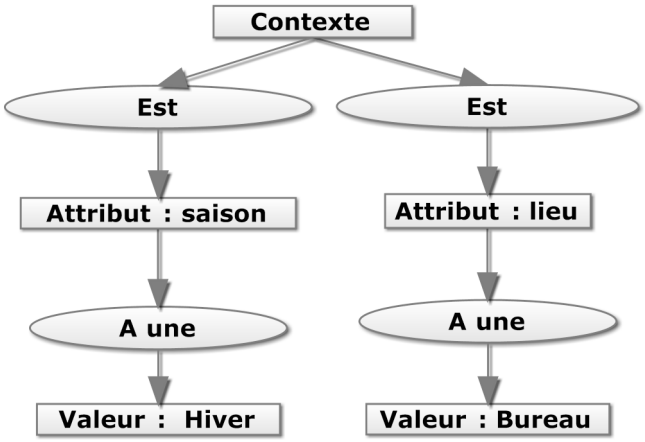

FIGURE 4.7 – Exemple d'un graphe conceptuel représentant un contexte utilisateur

Exemple : La figure 4.8 représente un service Web Banque noté (WS_banque) qui est composé d'une opération retrait et un paramètre solde qui est un paramètre sensible.

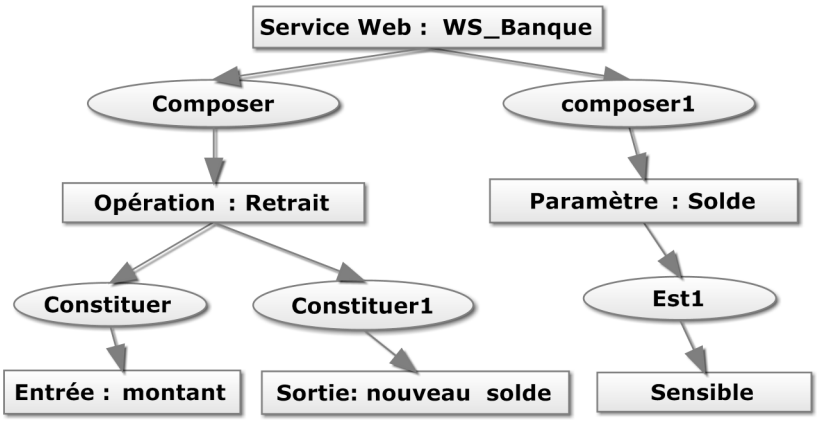

FIGURE 4.8 – Exemple d'un graphe conceptuel représentant un service Web simple

4.5.2.6 Représentation de la requête du sujet

La requête du sujet est un graphe conceptuel de type headed graphe dont le concept (Requête) représente la tête sémantique. Le concept (Service Web) correspond au service Web simple invoqué par le sujet, les opérations et les paramètres demandés par le sujet sont représentés respectivement par les concepts (Opération, Paramètre). L'option de composition est représentée par le concept (option) et l'action demandée à effectuer sur les opérations ou paramètres par le sujet est représentée par le concept (action).

Exemples : La figure 4.9 représente un exemple d'une requête composite, la requête est destinée à 3 services Web. (WS_Banque) est un service Web mis à la disponibilité des clients d'une banque afin que ces derniers fassent des consultations de compte, des retraits ...etc. Le service Web (WS_voyage) propose des services pour consulter, organiser,

réserver des voyages où le payement peut se faire directement par Internet. (WS_Hotel) est un service Web d'un hôtel qui propose des brochures, des réservations de chambres ...etc. Dans cet exemple, le sujet veut faire une réservation d'un voyage à partir du service Web voyage, faire un retrait sur son compte à partir du service Web banque pour payer la réservation et consulter les tarifs des chambres proposés par le service Web hôtel.

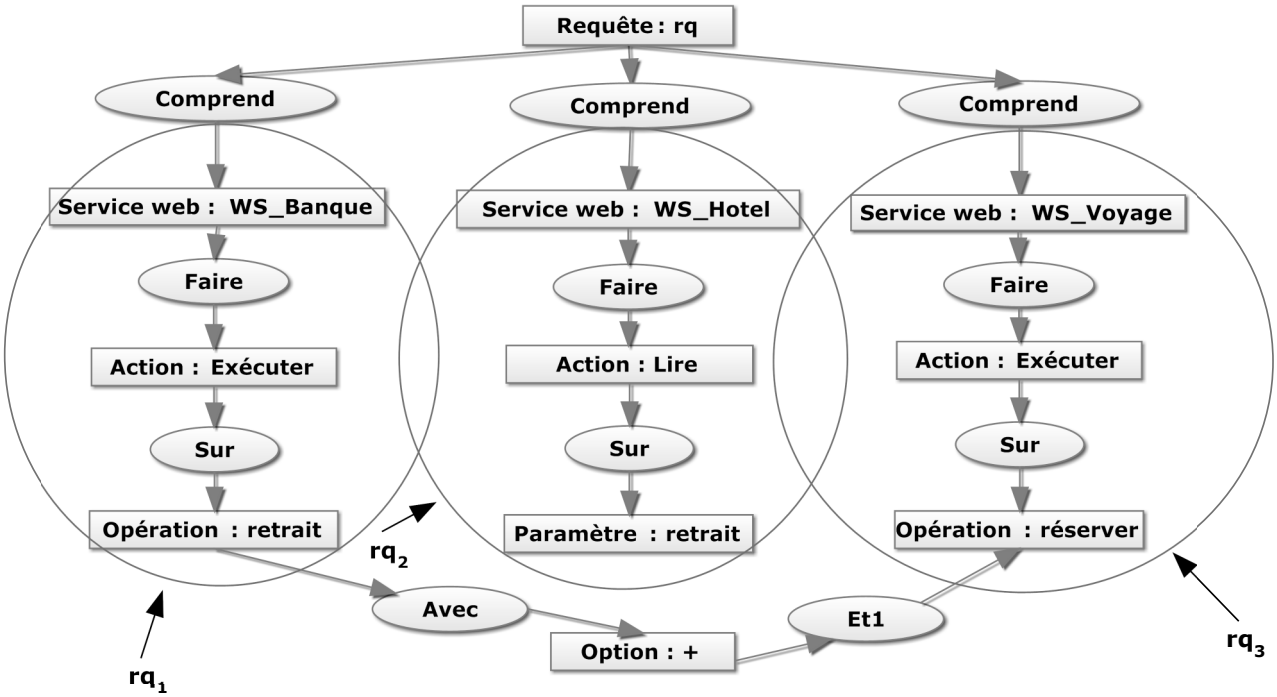

FIGURE 4.9 – Exemple d'un graphe conceptuel représentant une requête

4.5.2.7 Représentation de la politique de contrôle d'accès

La politique de contrôle d'accès est représentée par des règles de graphe, chaque rôle contient un ensemble de règles où chaque règle contient une hypothèse et un résultat (conclusion). Les hypothèses contiennent le rôle affecté au sujet ainsi que les contextes utilisateurs (dans le cas où ces derniers interviennent dans la condition d'accès). Le résultat contient le rôle, les permissions d'accès affectées à ce rôle et les actions sur les opérations et paramètres.

Exemple : La figure 4.10 illustre un exemple d'une règle de graphe. Dans cet exemple, l'hypothèse signifie qu'un sujet dans son lieu de travail (Bureau) avec un rôle administrateur a la permission d'écrire (Write) sur l'opération gestion et de lire (Read) le paramètre solde. La condition dans son lieu de travail est représentée par le contexte utilisateur avec l'intitulé (Lieu) qui représente le lieu actuel du sujet et la valeur avec un marqueur (bureau) qui signifie que le sujet est dans son bureau (lieu de travail). Dans la conclusion

(Permission : Permit) signifie que la permission est accordée, dans le cas où la permission est refusée, la permission aura un marqueur (Deny).

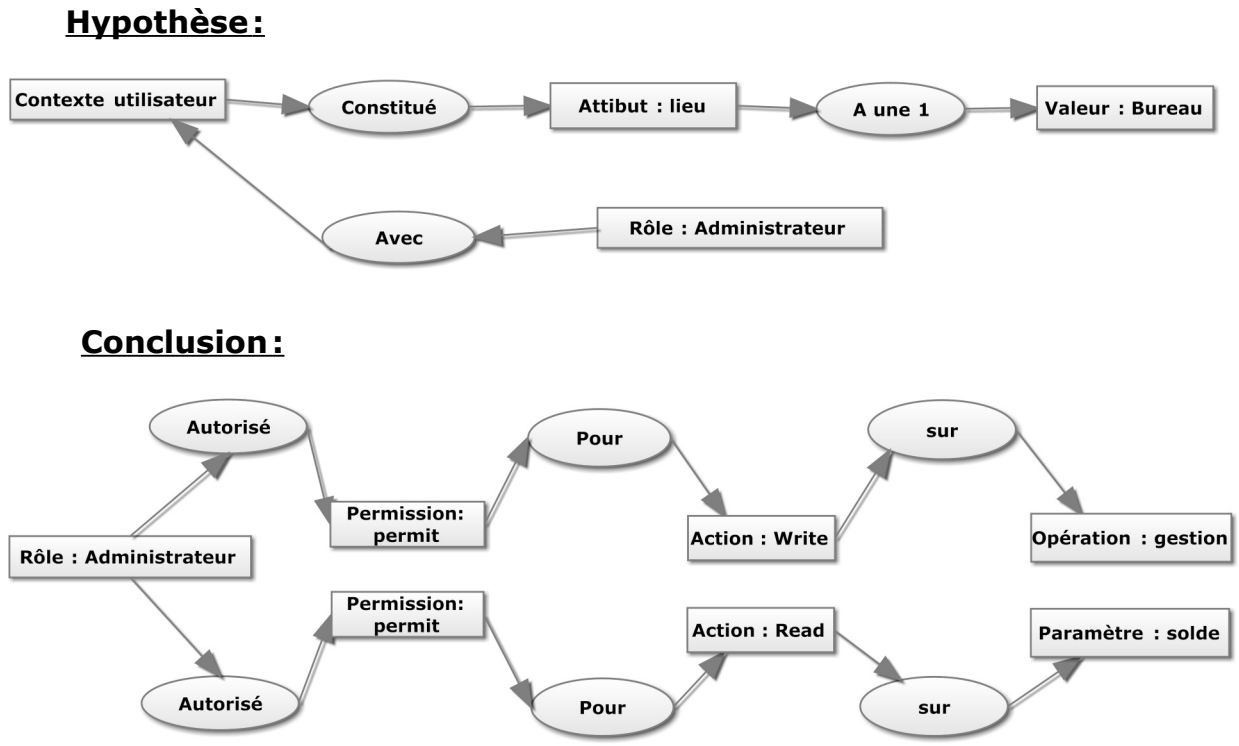

FIGURE 4.10 – Exemple d'un graphe conceptuel représentant une règle

4.6 Contrôle d'accès aux services Web composites (AC-CWS)

Dans cette partie nous représentons les différentes étapes du processus de contrôle d'accès d'AC-CWS. Ce processus est entièrement modélisé par les graphes conceptuels où les différents concepts sont modélisés par des graphes conceptuels et les différentes étapes du processus de contrôle d'accès sont réalisées par les opérations sur les graphes conceptuels.

Afin de mieux comprendre le processus de contrôle d'accès de AC-CWS, nous présenterons en premier les principales opérations sur les graphes utilisés dans le processus puis nous présenterons le processus de contrôle d'accès.

4.6.1 Les opérations sur les graphes conceptuels

L'un des principaux avantages qu'offrent les graphes conceptuels vient du fait que des raisonnements peuvent être effectués sur les connaissances représentées. Ces raisonnements

peuvent être vus comme des opérations de graphes et peuvent ainsi se baser sur les travaux de la théorie des graphes, puisque les graphes conceptuels sont des graphes.

4.6.1.1 La projection

L'opération de projection est un morphisme de graphes autorisant la restriction des étiquettes des sommets, conformément au support. Elle permet de tester la relation de spécialisation entre graphes conceptuels, en déterminant si un graphe conceptuel est plus spécifique qu'un autre. Cette opération est utilisée pour l'interrogation de bases de connaissances dans le modèle des graphes conceptuels.

Formellement, la projection est définie comme une application "Π" des n œuds d'un graphe "H" vers les n œuds d'un graphe "G" tel que :

- Pour chaque concept "c" de "H", $\Pi(c)$ (projection ou image du concept c) est soit une spécialisation de c, soit identique à c ;
- Pour chaque relation r de H, " $\Pi(r)$ " (projection ou image de la relation r) est soit une spécialisation de r, soit identique à r.
- Si le i^{eme} arc de "r" est lié à un concept "c" dans "H", alors le i^{eme} arc de "$\Pi(r)$" doit être lié à "$\Pi(c)$" dans "G".

Une projection d'un graphe "H" sur un graphe "G" n'est pas toujours unique : le graphe "G" peut avoir plusieurs sous-graphes dont il existe une projection de "H" sur ce sous-graphe vérifiant la condition de la projection. La figure 4.11 montre un exemple de projection d'un graphe d'un graphe (Rs) sur un graphe (R).

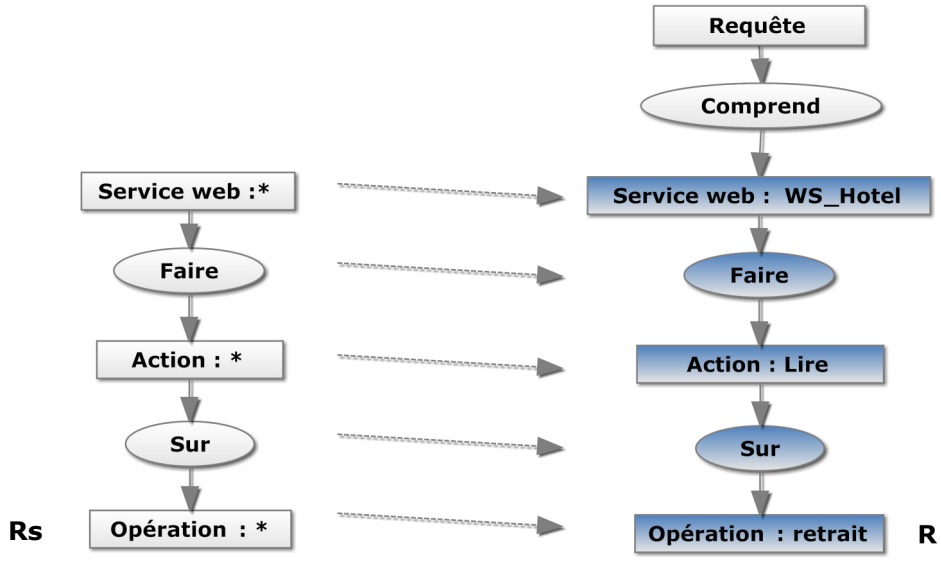

FIGURE 4.11 – Opération de projection

L'opération de projection nous permet de déterminer directement si un graphe conceptuel est plus spécialisé qu'un autre graphe c'est-à-dire si G est un sous graphe de H.

4.6.1.2 La normalisation

La normalisation [60] est l'opération qui rend un graphe sous forme normale.

Un graphe sous forme normale respecte une structure où les référents sont uniques : elle est obtenue en fusionnant les n œuds concepts ayant le même marqueur individuel. Ainsi, lorsqu'on utilise la forme normale, il ne peut exister deux appellations différentes pour référer à une même instance d'un certain concept dans un même graphe.

D'une manière formelle, considérons "H" un graphe conceptuel, et "C" l'ensemble de ces concepts. "H" est en forme normale si pour n'importe quel couple de concepts "$c1$" et "$c2$" appartenant à "C", référent ($c1$) différent du référent ($c2$).
La figure 4.12 montre un exemple de normalisation du graphe R.

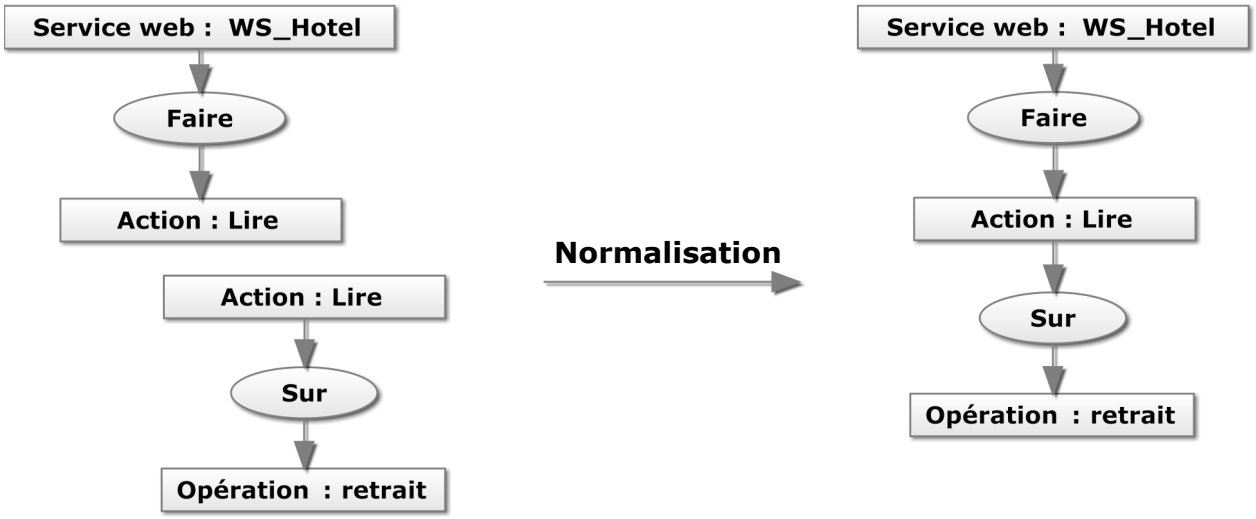

FIGURE 4.12 – Opération de normalisation

L'objectif de la normalisation d'un graphe est d'éviter des ambiguïtés sémantiques et logiques dans les graphes conceptuels. Sans la forme normale, deux graphes peuvent avoir le même sens et la même interprétation logique sans avoir le même graphe qui les représente [84].

4.6.1.3 La somme disjointe :

La somme disjointe de deux graphes conceptuels H et C est le graphe conceptuel (non connexe) HC constitué par la juxtaposition de H et C.
Dans la figure 4.13, le graphe conceptuel HC est obtenu à partir des graphes conceptuels H et C par somme disjointe. La somme disjointe nous permet de regrouper deux graphes

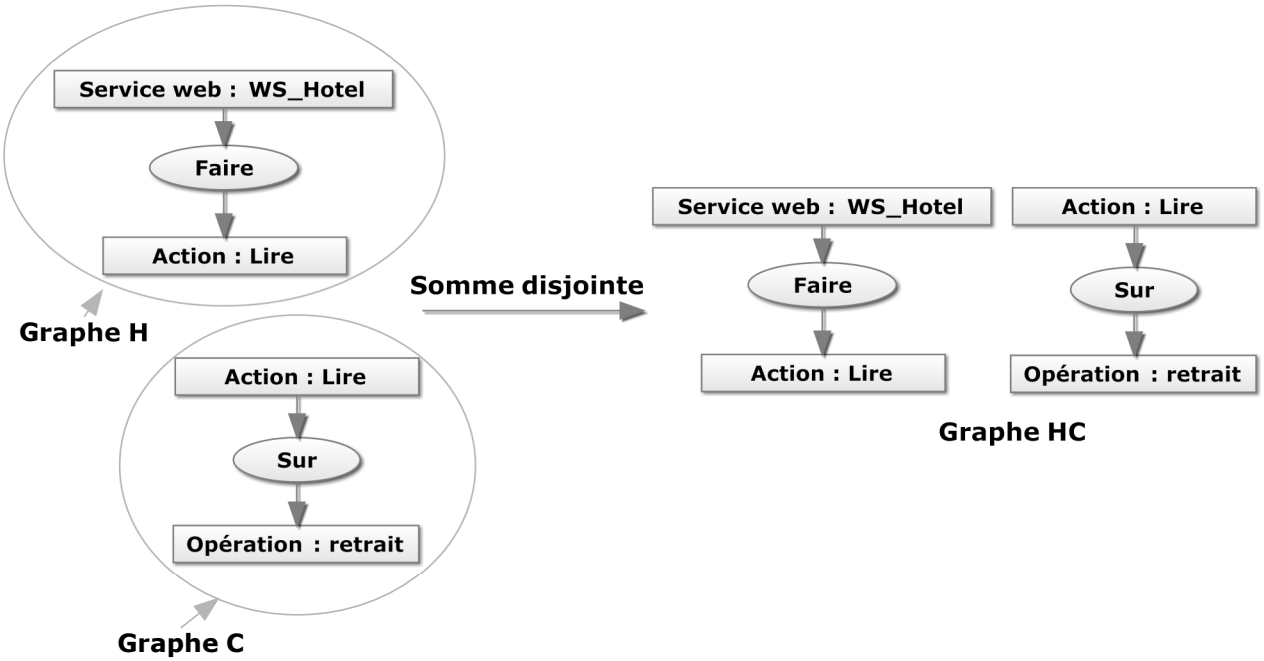

FIGURE 4.13 – Somme disjointe

dans un seul graphe afin d'utiliser qu'un seul graphe.

4.6.2 Les étapes du processus d'AC-SWC

Dans un environnement distribué, où chaque politique de contrôle d'accès étant conçu par des fournisseurs de service Web différents, les politiques de contrôle d'accès sont autonomes et hétérogènes. L'hétérogénéité se situe au niveau de l'interprétation du monde réel. Les conflits résultent de l'utilisation de domaines de données différents selon le domaine d'application. En effet, des données similaires peuvent être représentées et interprétées différemment dans chaque politique. L'utilisation des cercles de confiances permet seulement de réduire la complexité en hétérogénéité. Masquer l'hétérogénéité revient à faire communiquer les services Web via une connaissance commune qui permet d'expliciter et de préciser le sens des données pour être interprétée correctement par différents systèmes. Cette connaissance peut être capturée grâce à des ontologies.

Dans le modèle proposé, deux types d'ontologie sont définies :

1. **L'ontologie de domaine du modèle de contrôle d'accès :** cette ontologie représente le vocabulaire utilisé par les différents services Web pour exprimer les

permissions d'accès, et les actions ainsi que les relations qui les lient. La figure 4.14 illustre un exemple d'une telle ontologie.

2. **L'ontologie de domaine dans un cercle de confiance :** cette ontologie représente le vocabulaire utilisé pour les différents attributs, opérations et paramètres ainsi que les différentes relations qui lient ces concepts. La figure 4.15 illustre une telle ontologie.

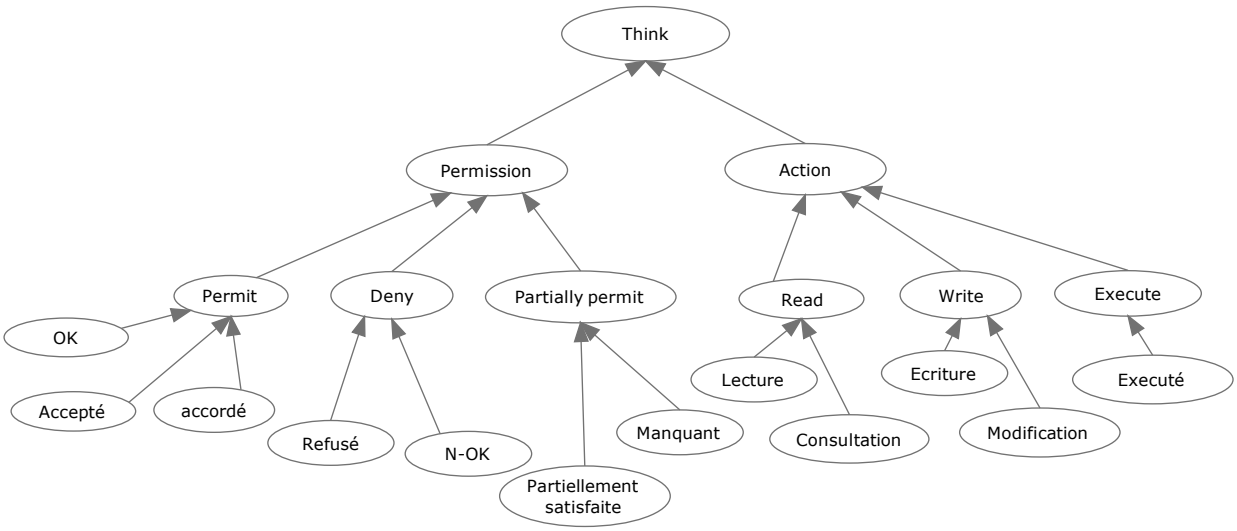

FIGURE 4.14 – Exemple d'une ontologie de domaine pour les permissions

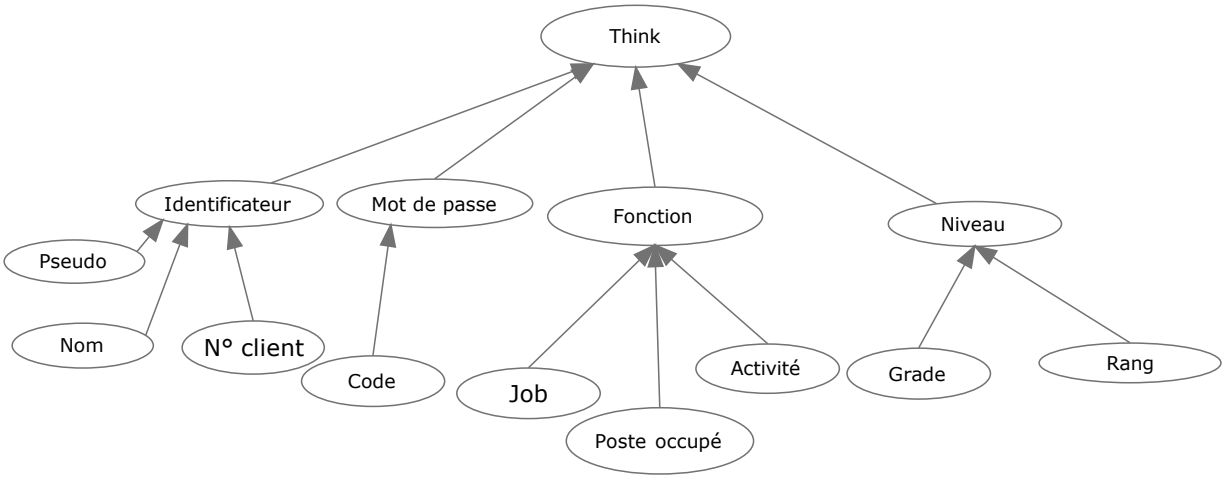

FIGURE 4.15 – Exemple d'une ontologie d'un cercle de confiance

La première étape dans le modèle de contrôle d'accès est la décomposition de la requête rq envoyée par le client en sous-requêtes (rq_i), où la décomposition se fait selon le service Web qui va vérifier et satisfaire la sous-requête. Ensuite, chaque sous-requête (rq_i) est

envoyée aux services Web concernés où chaque service Web vérifie la permission d'accès de cette dernière.

Dans un cercle de confiance utilisant la politique de contrôle d'accès du modèle proposé, le service Web construit en premier le profil du sujet en utilisant l'historique du sujet et l'ontologie de domaine du cercle de confiance, puis à partir du profil du sujet, attribuer un rôle au sujet et vérifier les permissions d'accès de la sous-requête (rq_i) et enfin envoyer le résultat au modèle de contrôle d'accès.

Une fois que toutes les permissions d'accès des différentes sous-requêtes (rq_i) reçues, celles qui sont issues d'un cercle de confiance différent du cercle de confiance qui utilise la politique de contrôle d'accès du modèle proposé seront adaptées à notre politique en utilisant l'ontologie de domaine du modèle de contrôle d'accès, ensuite résoudre le conflit en utilisant les option de composition et dans le cas où le résultat de la requête (rq) est ($p - permit$) (partiellement acceptée) un négociation avec le sujet sera lancer, afin de compléter les données manquantes. Puis enfin, envoyer la permission d'accès à la requête (rq) au sujet. Les différentes étapes du processus de contrôle d'accès utilisant les graphes conceptuels sont représentées par la figure 4.16.

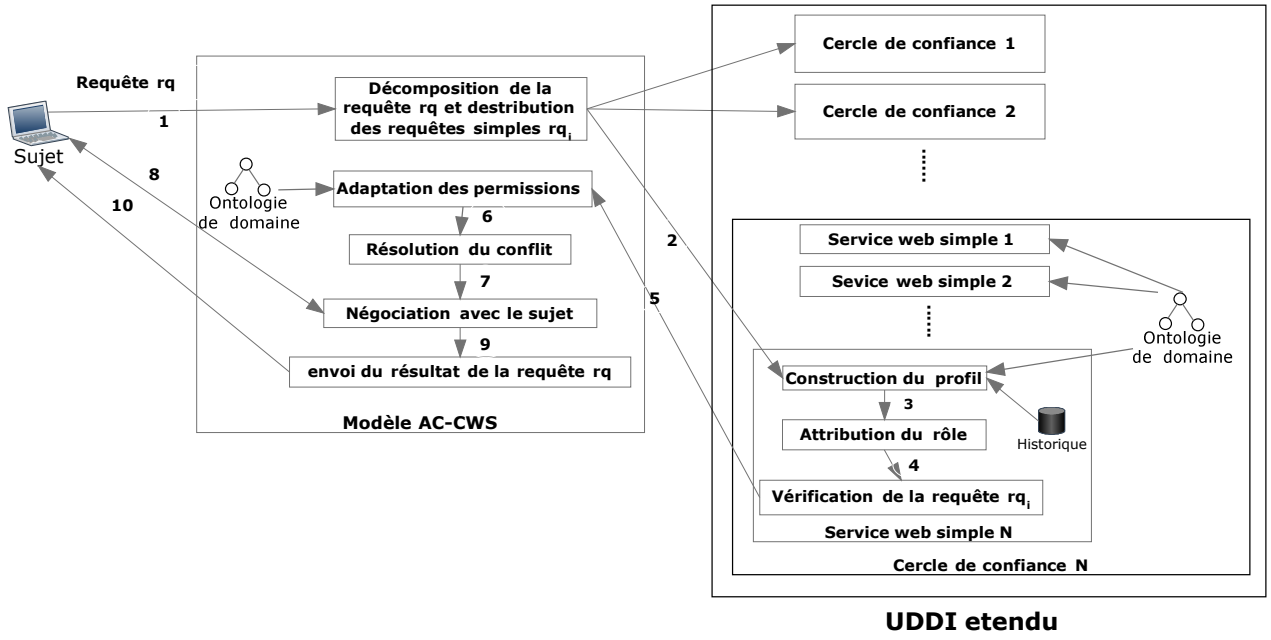

FIGURE 4.16 – Étapes d'exécution du modèle AC-CWS

L'historique contient l'ensemble des données récoltées par le service Web simple sur l'ensemble de ses utilisateurs, ces données sont par exemple (Le nombre de connexions, les préférences de l'utilisateur... etc.).

Le processus d'AC-WSC est composé de deux parties, la parties traiter par le modèle de contrôle d'accès et la partie qui traitée au niveau des services Web simples, la deuxième partie est traitée en parallèle. Les étapes de la première partie sont :

1. Décomposition de la requête rq en sous-requête rq_i ;
2. Distribution des sous-requêtes rq_i ;
3. Adaptation des permissions d'accès (cette étape est exécutée dans le cas où des services Web appartiennent à un cercle de confiance qui n'utilise pas la politique de contrôle d'accès proposé) ;
4. Résolution du conflit (cette étape est exécutée une fois que toutes les réponses aux requêtes simples sont reçues)
5. Négociation avec le sujet (pour adapter la requête du sujet) ;
6. Envoi du résultat de la requête.

Les étapes de la deuxième partie sont :

1. Construction du profil ;
2. Attribution du rôle ;
3. Vérification des permissions d'accès.

4.6.2.1 Décomposition de la requête et distribution des sous-requêtes

Soit (rq) un requête composite et (rs) un graphe conceptuel qui représente les concepts génériques et les relations d'une requête simple, le processus de décomposition de la requête (rq) (voir la figure 4.17) suit les étapes suivantes :

1. Normalisation de la requête (rq) ;
2. Projection de (rs) sur (rq) ;
3. Sélection des sous-requêtes (rq_i) résultants de la projection ;
4. Distribution des sous-requêtes (rq_i) aux services Web concernés.

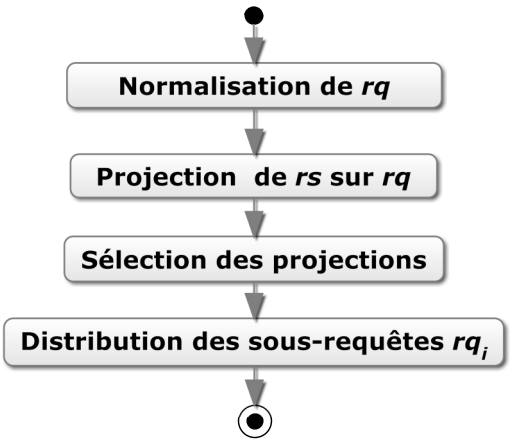

FIGURE 4.17 – Processus de décomposition de la requête

4.6.2.2 Construction du profil

Le processus de construction du profil est la première étape effectuée par le service Web simple qui va vérifier la requête simple (rq_i). Les attributs du sujet, l'historique du sujet ainsi que l'ontologie de domaine seront nécessaires pour la réalisation de ce processus. Soient $(s, sa, pr, hetpt)$ des graphes conceptuel qui représentent respectivement (Sujet, Prototype d'attribut du sujet, Profil, Historique du sujet et Profil temporaire) Les différentes étapes du processus (voir la figure 4.18) sont les suivantes :

1. Normalisation de (s) ;
2. Récupération de l'historique du sujet (h) (les données supplémentaires sur le sujet) ;
3. Projection de (sa) sur (s) pour la récupération des attributs du sujet ;
4. Sélection des résultats de la projection ;
5. Faire une somme disjointe entre les résultats sélectionnés de la projection avec (pt) et (h) avec (pt) en utilisant l'ontologie de domaine du cercle de confiance auquel appartient le service Web simple ;
6. Normalisation du résultat de la somme disjointe (pt) ;
7. Projection de (pt) sur chaque (p) ;
8. S'il y a une projection alors sélection du graphe résultant de la projection ;
9. Sinon sélection du graphe du profil invité.

4.6.2.3 Attribution du rôle

Les permissions d'accès ne sont pas affectées directement aux sujets, mais via un rôle attribué au sujet, l'attribution du rôle dépend du profil sujet résultant du processus de construction du profil. Soient (r), (rt) les graphes conceptuels représentant respectivement

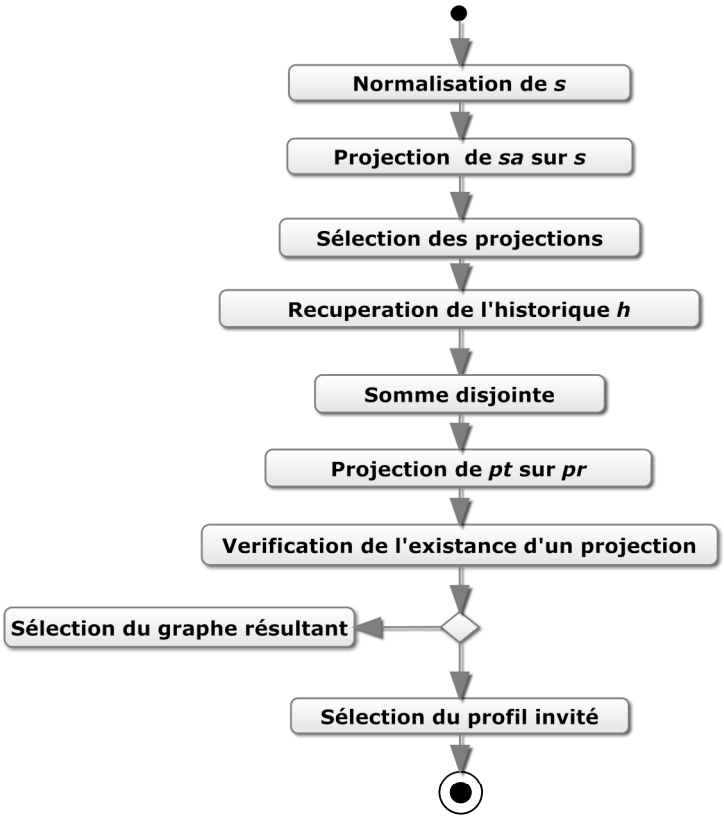

FIGURE 4.18 – Processus de construction du profil

l'affectation des profils aux rôles. Les étapes du processus d'attribution du rôle (Voir lfigure 4.19) sont les suivantes :

1. Sélection du concept profil dans le graphe conceptuel (p) de l'étape précédente.
2. Construction du graphe (rt), ce graphe est composé d'un concept (rôle) générique et le concept (profil) sélectionné lié par la relation (contient) (voir la figure 4.20) ;
3. Projection de (rt) dans (r) ;
4. Sélection du concept (Rôle) du graphe résultant.

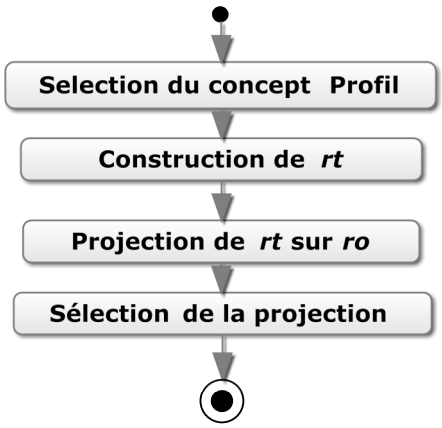

FIGURE 4.19 – Processus d'attribution du rôle

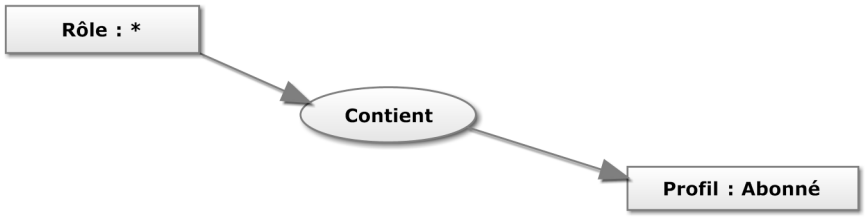

FIGURE 4.20 – Exemple d'un graphe conceptuel Rt

4.6.2.4 Vérification des permissions d'accès

La vérification des permissions d'accès est le processus le plus important, car c'est au niveau de cette étape que les décisions aux requêtes simples sont prises. Les étapes du processus de vérification des permissions d'accès (voir figure 4.21) sont les suivantes :

1. Sélection des contextes utilisateurs nécessaires à la vérification des permissions ;
2. Sélection des règles applicables du rôle avec les contextes utilisateur ;
3. Projection des requêtes simples sur les conclusions des règles applicables ;
   (a) S'il existe une seul permission Alors sélection du résultat de la projection ;
   (b) S'il existe plusieurs permissions Alors sélection de la permission dont l'hypothèse de la règle contient un contexte.
4. S'il n'existe pas de permission et que l'opération et le paramètre demandé n'est pas sensible Alors parcourir les rôles hérités ;
   (a) S'il n'existe pas de permission Alors la permission est refusée ;
   (b) S'il existe plusieurs permissions Alors résolution de conflit interne ;
   (c) S'il existe une seul permission Alors sélection de la permission résultant de la projection.
5. Envoi du résultat.

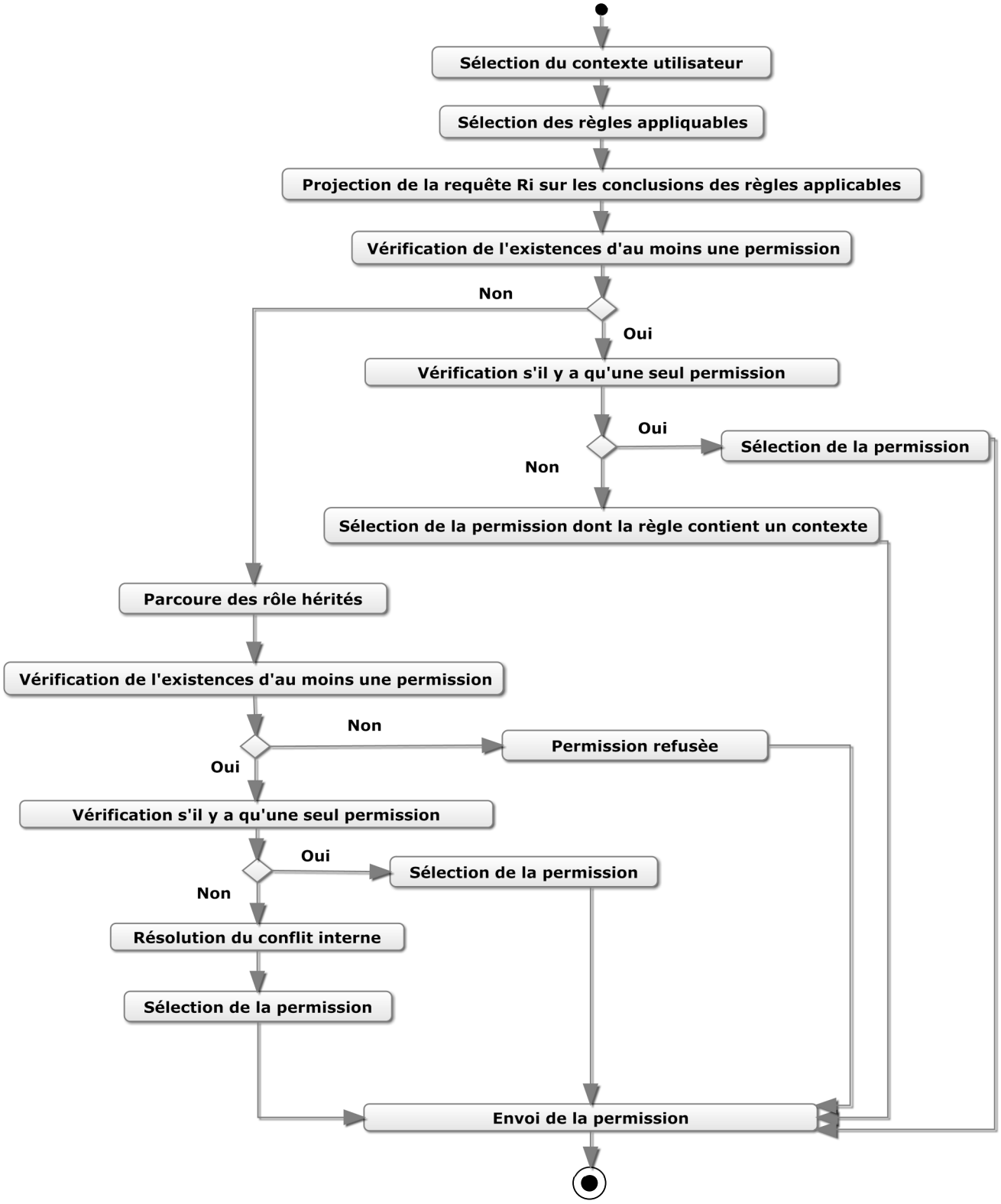

FIGURE 4.21 – Processus vérification de la requête simple

4.6.2.5 Adaptation des permissions d'accès

Une fois que toutes les permissions d'accès aux requêtes (rq_i) reçues, les permissions issues des services Web appartenant à des cercles de confiance qui n'utilisent pas notre

politique de contrôles d'accès seront d'abord adaptées en utilisant l'ontologie de domaine du modèle proposé puis, à partir du résultat de l'adaptation, construire le graphe conceptuel correspondant à la permission d'accès. Les étapes du processus d'adaptation des permissions d'accès (voir la figure 4.22) sont les suivantes :

1. Réception des différentes permissions des requêtes simples (rq_i) ;
2. Vérification pour chaque permission d'une requête (rq_i), si elle n'est pas issue d'un cercle de confiance qui utilise notre politique ;
3. Mapping de la permission avec l'ontologie du domaine ;
4. Construction du graphe conceptuel correspondent.

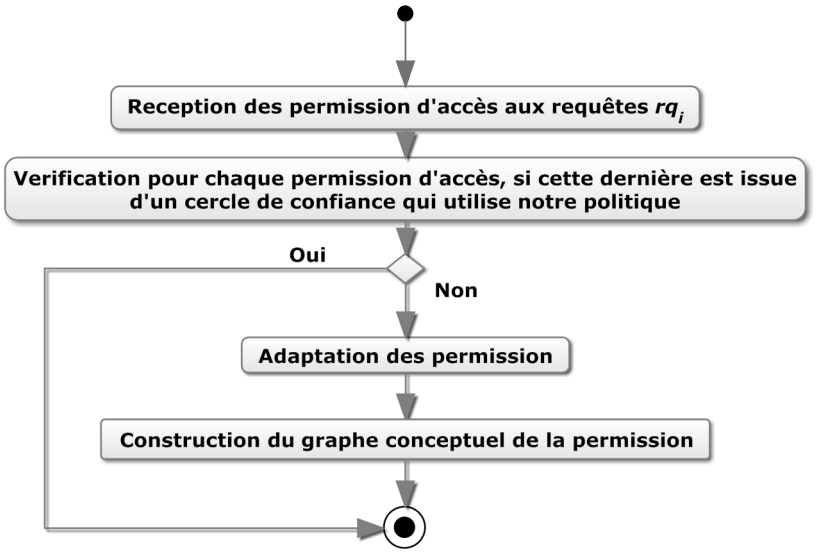

FIGURE 4.22 – Processus adaptation des permissions

4.6.2.6 Résolution du conflit

Les permissions d'accès sont représentées sous forme de graphe conceptuel, dans cette étape du processus, nous nous intéressons aux résultats des permissions d'accès. À partir des permissions d'accès des différents services Web collaborant pour la satisfaction de la requête et les options de composition, on construit une fonction logique telle que les variables sont les permissions (une permission $permit = 1$, et une permission $deny = 0$) et les opérateurs logiques sont les options de composition représentées par le tableau 4.1, ainsi le résultat de la fonction logique représente le résultat de la résolution du conflit. La résolution du conflit intervient en cas de présence de permissions d'accès acceptées (*permit*) et refusées (*deny*) au même temps. La vérification de la permission d'accès de la requête (rq) se fait à partir des permissions d'accès des requêtes simples (rq_i) et les options de composition entre les requêtes (rq_i). À l'issu de ce processus, une décision de

négociation avec le sujet est prise. En effet, la négociation avec le sujet est lancée dans le cas d'existence de permission d'accès partiellement satisfaite ($p-permit$) et que la réponse du sujet va changer la permission d'accès de la requête (rq). Les étapes de l'algorithme de résolution de conflit (voir la figure 4.23) sont les suivantes :

1. Vérification de l'existence des permissions partiellement satisfaites ($p-permit$) dans les permissions des (rq_i) ;
2. S'il n'existe pas Alors exécuter la composition de la requête (rq) et envoyer la permission ;
3. S'il existe ;
   (a) Remplacer chaque permission ($p - permit$) par ($permit$) dans la requête (rq) (pour vérifier la permission dans le cas où le sujet retourne tous les attributs manquants) ;
   (b) Vérification de la permission résultant de la composition ;
   (c) Si le résultat de la permission est refusé ($deny$) Alors envoyer la permission au sujet ;
   (d) Si le résultat de la permission est accepté ($permit$)Alors
      i. Remplacer chaque permission ($p - permit$) par ($deny$) dans la requête rq de départ (avant l'étape 3)(pour vérifier la permission dans le cas où le sujet ne retourne aucun attribut manquant) ;
      ii. Si la permission est acceptée Alors envoyer le résultat ;
      iii. Si la permission est refusée Alors lancer le processus de négociation,
      iv. Récupération du résultat du processus de négociation.

Permission 1	**Permission 2**	**Fonction** $f1$	**Fonction** $f2$
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

TABLE 4.1 – Table de vérité des différentes options de composition

tel que

- $f1 = permission1 \wedge permission2$: Est la fonction de l'option de composition en série ;
- $f2 = permission1 \vee permission2$: Est la fonction de l'option de composition au choix ;

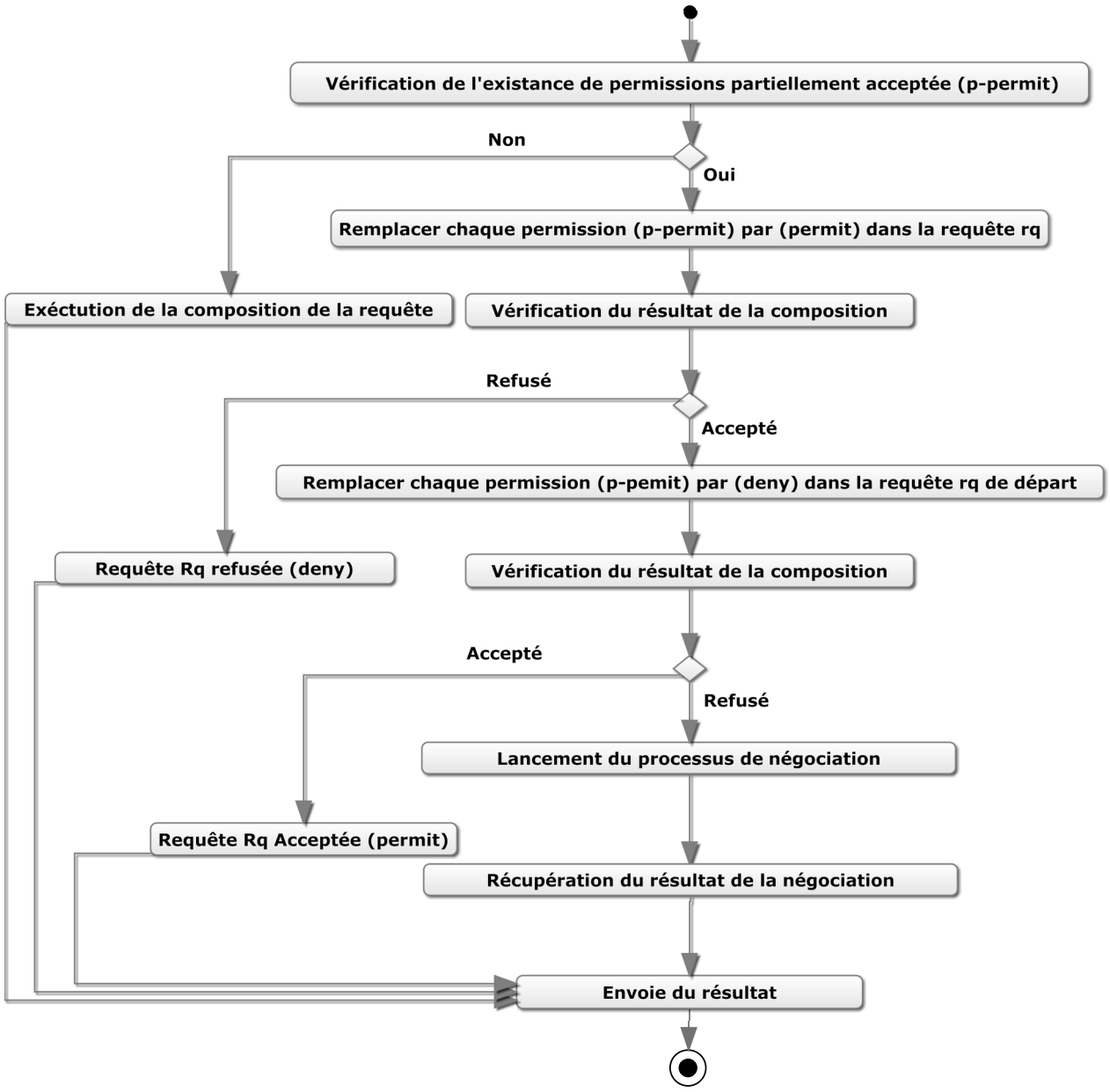

FIGURE 4.23 – Processus résolution du conflit

4.6.2.7 Négociation avec le sujet

Le processus de négociation est lancé seulement si les attributs manquants du sujet peuvent influencer sur la permission d'accès de la requête composite (rq) c'est-à-dire si le sujet envoi les attributs manquants, la permission d'accès sera accepté ($permit$) sinon si le sujet n'envoie pas les attributs manquants, la permission d'accès sera refusée ($deny$).
les étapes du processus de négociation (voir la figure 4.24) sont les suivantes :

1. Envoyer la liste des attributs manquants pour la satisfaction de la requête (rq) ;
2. Si le sujet retourne les attributs manquants Alors la requête (rq) est acceptée ($permit$) ;
3. Sinon si jusqu'à la fin de la session, le sujet n'a pas retourné la liste des attributs

manquant Alors la requête est refusée(*deny*).

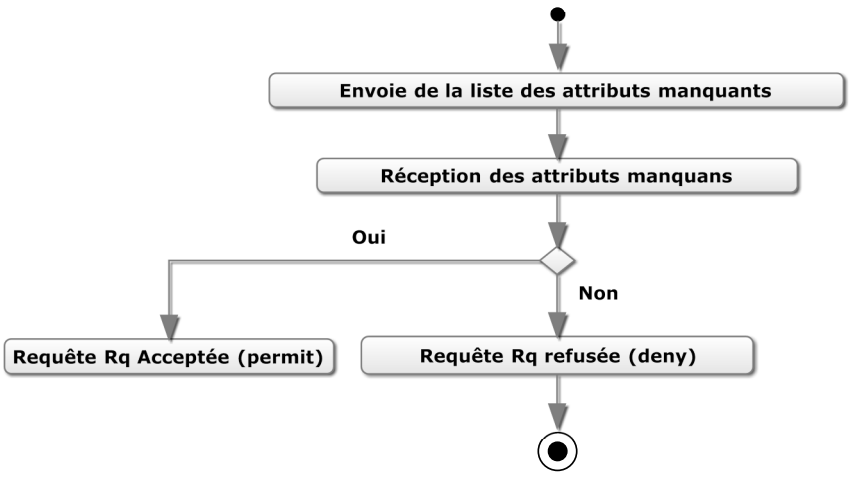

FIGURE 4.24 – Processus négociation

4.7 Positionnement de notre travail

Le modèle que nous avons proposé pour le contrôle d'accès aux services Web composites décrits sémantiquement se base sur le concept "rôle" du modèle RBAC [70] de base. L'enrichissement de la notion de rôle dans le modèle proposé réside dans l'affectation des rôles aux différents sujets. En effet l'affectation du rôle dans le modèle proposé se fait via le profil du sujet. Dans certain cas, le modèle proposé offre la possibilité au sujet de compléter leurs requêtes via une négociation, ce concept est similaire à celui proposé dans le modèle WS-AC1 [15]. Les permissions d'accès dans le modèle proposé dépendent non seulement du rôle activé par le sujet, mais aussi des contraintes imposées par le fournisseur du service Web, les contraintes sont représentées sous forme de contexte similaire au contexte du modèle C-TMAC [85], néanmoins le contexte dans le modèle proposé s'applique au niveau des permissions non pas au niveau du rôle. Le modèle de contrôle d'accès fourni deux types d'ontologies de domaine, l'ontologie de domaine au niveau du cercle de confiance qui contient le vocabulaire utilisé par les différents services Web et l'ontologie du domaine au niveau du modèle contrôle d'accès qui contient les différentes notations des permissions d'accès.

Dans la littérature, plusieurs travaux traitent ce problème, nous citons parmi eux : WS-XACML [4], WS-AC1 [15], X-RBAC [19], X-GTRBAC [18] et WS-RBAC [88], néanmoins, à l'exception du modèle WS-RBAC [88], tous les autres modèles supposent que la requête du sujet est destinée à un seul service Web.

Le concept rôle a été utilisé dans les modèles X-RBC [19], X-GTRBAC [18] et WS-RBAC [88], néanmoins, l'attribution des rôles aux utilisateurs se fait par un administrateur selon le modèle RBAC [70], cette méthode parait efficace pour les modèles de contrôle d'accès aux systèmes d'information, mais elle parait difficile voir impossible à appliquer dans l'environnement des services Web où le nombre d'utilisateurs est important.

Afin de formaliser le modèle proposé, nous avons utilisé les graphes conceptuels, à notre connaissance, l'utilisation des graphes conceptuels n'a pas été appliquée dans la formalisation d'un modèle de contrôle d'accès dans la littérature. Les différentes entités du modèle proposé sont représentées par les graphes conceptuels, et les différentes étapes du modèle de contrôle d'accès sont réalisées par les opérations sur les graphes conceptuels.

Dans la catégorie des modèles formels pour le contrôle d'accès, on trouve le modèle de Bertino [14], Jajodia [47] qui proposent chacun un modèle formel basé sur la logique des prédicats pour la représentation des politiques de contrôle d'accès, Agrawal [2] propose un modèle formel basé sur la logique de description afin de formaliser un modèle de contrôle d'accès aux services Web, koch [51] propose un modèle formel basé de graphe afin de représenter le modèle RBAC, Shafiq [71] propose un modèle formel à base sur les réseaux de Petri pour la vérification et la validation d'un modèle RBAC en temps réel et Baina [6] propose un modèle basé sur les automates temporisés pour représenter les opérations du modèle PolyORbac.

En termes d'expressivité, les modèles formels utilisant la logique de prédicat Jajodia [47], Bertino [14] et la logique de description Agarwal [2] sont plus expressifs que les graphes conceptuels. En effet, avec les graphes conceptuels on ne peut pas exprimer la négation, par exemple une permission permise (permit) est le contraire d'une permission interdite (deny). Notre langage n'exprime pas aussi d'une manière exacte l'opérateur $\forall$. D'un autre côté, les graphes conceptuels permettent de représenter des relations de différentes aritès à la différence de la logique qui permet de représenter que des relations binaires.

En termes de facilité d'utilisation, un graphe édité à l'écran par le biais d'une interface conviviale n'a pas, auprès d'utilisateurs non informaticiens, le côté rebutant que peuvent avoir les formules logiques des logiques de description et la logique de prédicat. Et un graphe conceptuel est plus clair et représentatif qu'un graphe [51] en réseaux de Petri [71], vu que les graphes conceptuels sont proches du langage naturel.
La présence d'une hiérarchie de types de relations se prête bien à la représentation des relations de spécialisation/généralisation.

Les modèles qui se base sur la logique des prédicats [47, 14], les graphes [51], les

réseaux de pertri [71], et les automates temporisés [6] ne fournissent pas de mécanisme pour raisonner sur les ontologiques, par contre la logique de description et les graphes conceptuels fournissent des mécanismes pour le raisonnement sur les ontologies. Mais l'inconvénient des graphes conceptuels est qu'il n'existe pas d'outils qui implémentent ce raisonnement contrairement à la logique de description, néanmoins, il exit un outil qui traduit une ontologie décrite en logique de description en ontologie décrite en graphes conceptuels intitulés TooCom [3].

4.8 Contribution

Nos principales contributions se situent dans :

- L'utilisation du profil comme intermédiaire entre le sujet et le rôle, permet aux fournisseurs de services Web de gérer automatiquement les attributions des rôles aux sujets ;
- L'utilisation de la sémantique qui permet de masquer l'hétérogénéité au niveau des cercles de confiances et permet l'adaptation des permissions d'accès au niveau du modèle de contrôle d'accès ;
- La décentralisation du contrôle d'accès fourni une meilleure sécurité et réduit le temps de réponse ;
- L'utilisation des graphes conceptuels comme formalise qui fournit les outils nécessaires à la représentation des différentes entités du système et des mécanismes de raisonnement sur ces dernières.

4.9 Conclusion

Dans ce chapitre, nous avons présenté un modèle de contrôle d'accès aux services Web décrit sémantiquement. Le modèle proposé est un modèle de contrôle d'accès distribué, où la vérification des permissions d'accès implique la participation des services Web simples intervenants dans la requête composite. Le modèle proposé se base sur le concept du rôle qui facilite la gestion des sujets, du concept profil qui fournit une gestion automatique des attributions des rôles aux sujets et la notion du contexte utilisateur qui réduit le nombre de permissions à affecter. L'utilisation de la sémantique rend le modèle flexible et réduit l'hétérogénéité engendrée par la diversité des politiques de contrôle d'accès.

Afin de formaliser le modèle proposé, nous avons utilisé le modèle des graphes conceptuels qui est un formalisme qui décrit les connaissances d'une manière graphique expressive, facile à manipuler et lisible. Les graphes conceptuels dans le modèle proposé sont utilisés afin de formaliser les différents éléments du modèle et de raisonner sur ces derniers.

3. a Tool to Operationalize an Ontology with the Conceptual Graph Model

Le prochain chapitre est réservé à la validation expérimentale de notre proposition.

Chapitre 5

Mise en œuvre

5.1 Introduction

Afin de valider la proposition, nous avons implémenté un prototype conformément à l'architecture des services Web, et nous avons réalisé quelques scénarios d'exécution que nous présenterons dans ce chapitre.

5.2 Outils de développement utilisés

Il existe un certain nombre d'outils qui implémentent les graphes conceptuels notamment pour une utilisation orientée recherche et extraction d'informations, nous citons : CoGITaNT [1] [75], CoGui [2] [76], Amine [73], CharGer [74] et bien d'autres. Cependant, très peu de ces outils offrent un environnement logiciel complet pour l'utilisation la plus large possible du modèle des graphes conceptuel, le stockage et la manipulation d'un grand nombre de graphes.

Pour ces raisons, nous utilisons le logiciel de création graphique de graphes conceptuels CoGui et une bibliothèque de raisonnement permettant de faire des opérations de graphes CoGITaNT, sous le système d'exploitation linux.

5.2.1 CoGui

CoGui est un outil visuel pour construire des bases de connaissances de graphes conceptuelles. Il permet de créer une base de connaissances, éditer sa structure et son contenu et de la contrôler. La base de connaissances peut être sérialisée en format XML appelé CoGXML [3]. De plus, un mécanisme basique de question/réponse a été développé pour

1. Conceptual Graphs Integrated Tools allowing Nested Typed graphs
2. Conceptual Graphs user interface
3. Les graphes conceptuels au format XML

des buts éducatifs ; c'est-à-dire afin d'illustrer comment une base de connaissances peut être requise et exploitée. Pour les mécanismes de question/réponse complexes et d'autres tâches de raisonnement, CoGITaNT doit être utilisé.

5.2.2 CoGITaNT

L'open source CoGITaNT est un ensemble d'outils logiciels permettant le développement d'applications basées sur le modèle des graphes conceptuels, développés par l'équipe «Représentation de connaissances par des graphes » de LIRMM [4].

La bibliothèque CoGITaNT est un ensemble de classes C++ permettant une gestion facile des graphes conceptuels ainsi que les objets du modèle comme le support et les règles. Pour chaque objet du modèle, est associé une classe dans CoGITaNT, et les structures de données utilisées sont des implémentations de base des objets du modèle (par exemple, un graphe est un ensemble de n œuds et d'arcs), de cette façon, il est possible à l'utilisateur connaissant le modèle des graphes conceptuels à comprendre la structure de CoGITaNT, et les extensions sont simplifiées.

5.2.3 Formats de représentation supportés

CoGITaNT est l'outil accompagnant CoGui. Il propose plusieurs formats qui lui permettent de manipuler les graphes en lecture et en écriture. Les fichiers peuvent être au format propre BCGCT [5] ou au format CoGXML propre à CoGui.

1. Le format BCGCT est le format natif de CoGITaNT. Il permet de représenter sous une forme lisible les différents objets manipulés par la bibliothèque :
   - Un support, formé d'un ensemble partiellement ordonné des types de concepts, des types de relations, des types d'emboitements [6], des signatures des types de relation et de la relation de conformité.
   - Des graphes conceptuels (connexes ou non connexes, simples ou emboités).
   - Et des règles.
2. Le format CoGXML permet de représenter des graphes conceptuels sous la forme de documents XML. Ce format standard permet ainsi d'échanger avec d'autres applications de manière assez aisée. En effet, le langage XML est un langage de représentation de données adapté pour décrire des données structurées. La DTD [7] qui a été définie permet la représentation des supports, des graphes et des règles

4. Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier
5. Base de connaissances Graphes Conceptuels Textuelle
6. Les types emboités sont une extension des graphes conceptuels simples, cette notion n'est pas utilisée dans ce travail.
7. Définition de Type de Document.

sous une forme qui peut facilement être interprétée par des hommes et par des programmes.

5.2.4 TooCoM

TooCoM [8] [78] est dédié à l'édition et à l'utilisation d'ontologies de domaine. Cet outil, fondé sur le modèle des Graphes conceptuels et s'appuyant sur la plate-forme de manipulation de graphes conceptuels CoGITaNT, permet de définir les primitives conceptuelles (c.-à-d. les Concepts et les Relations) et la sémantique formelle du domaine considéré (c.-à-d. les Schémas d'Axiomes et Axiomes de Domaine) via une approche graphique. TooCom permet d'importer et d'exporter des ontologies au format OWL en utilisant l'API d'OWL.

TooCoM offre une interface permettant l'édition d'une ontologie, incluant le vocabulaire conceptuel et les axiomes, la spécification des contextes d'usage de chaque axiome et l'opérationnalisation de l'ontologie. Un moteur d'inférence basé sur CoGITaNT.
L'interface TooCoM est écrite en Java et communique avec un serveur CoGITaNT écrit en C++.

5.2.5 Serveur d'application GlassFish

GlassFish [9] est un serveur d'application prenant en charge la plate-forme Java Enterprise Édition 6 (Java EE 6) standard. GlassFish est utilisé dans notre application afin de déployer les différents services Web de notre prototype.

5.3 Architecture du système de AC-CWS

L'architecture du système proposé (figure 5.1) est basée sur l'architecture des services Web. Dans cette architecture le médiateur est un service Web qui traite les requêtes des applications clientes et communique avec les différents services Web afin de fournir une permission d'accès.

Les composants de cette architecture sont décrits comme suit :

- **Requête** rq **:** C'est la requête émise par l'application cliente destinée à un service Web composite ;
- **Sous-requête** rq_i **:** C'est une sous-requête résultant de la décomposition de la requête rq.
- **Application cliente :** C'est l'application qui émet la requête rq ;

8. a Tool to Operationalize an Ontology with the Conceptual Graph Model
9. http ://www.oracle.com/goto/glassfish

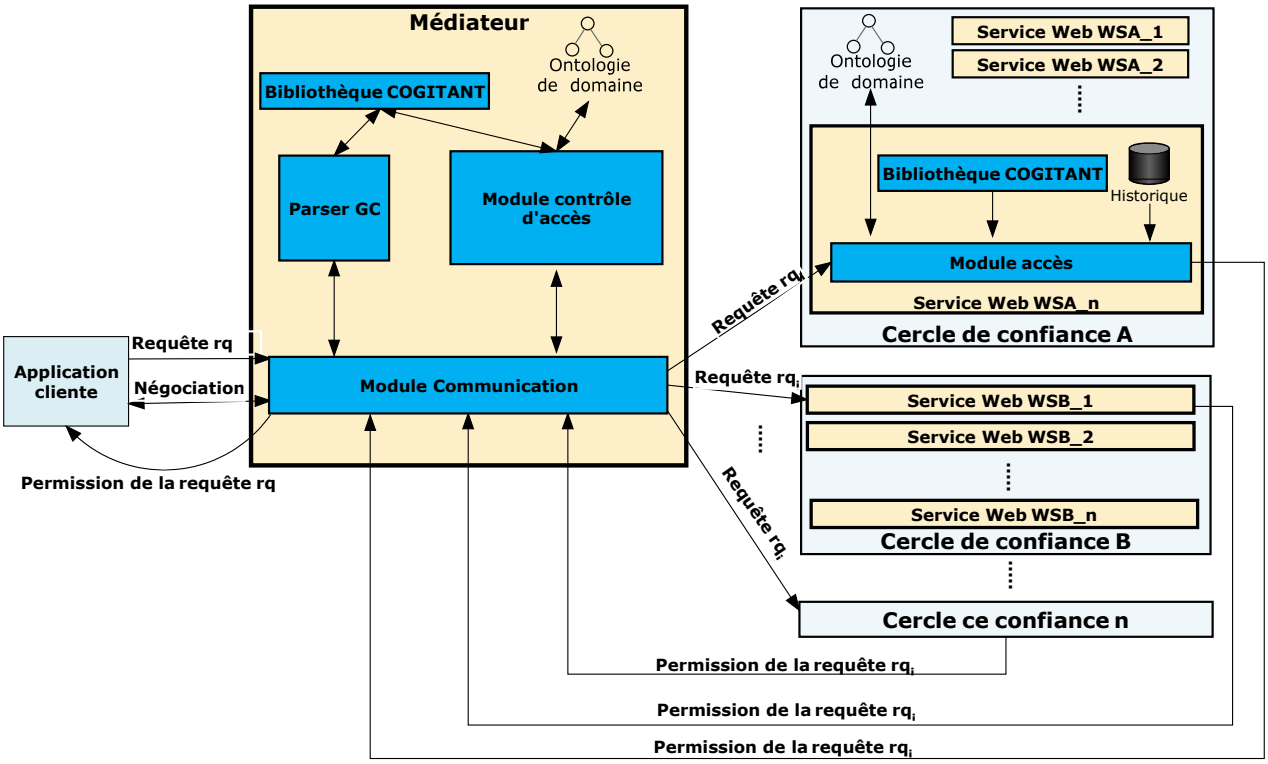

FIGURE 5.1 – Architecture du système proposé

- **Cercle de confiance :** Regroupe les services Web selon la politique de contrôle d'accès. Dans la figure 5.1, le cercle de confiance A regroupe les services Web qui utilisent la politique du modèle de contrôle d'accès proposé ;
- **Service Web :** C'est le service Web qui satisfait et vérifie la sous-requête rq_i ;
- **Médiateur :** C'est le service Web qui applique la politique de contrôle d'accès ;

5.4 Communication entre les différents composants du système

La communication entre les différents composants du système est réalisée par le protocole SOAP, qui est un protocole de transmission de messages basé sur XML. En effet le protocole SOAP est le mieux adapté pour le système vu que ce dernier est implémenté avec l'architecture des services Web.

L'application cliente représente un consommateur du service Web, elle émet un requête (rq) au médiateur sous le format XML via un message SOAP. Une fois décomposées, les sous-requêtes (rq_i) résultantes sont envoyées aux différents services Web par des mes-

sages SOAP. Les différentes permissions d'accès aux sous-requêtes (rq_i) sont retournées aux médiateurs via un message SOAP. Enfin, la permission d'accès à la requête (rq) est retournée à l'application cliente par un message SOAP. Le diagramme de séquence de la figure 5.2 représente les différentes interactions entre les différents composants.

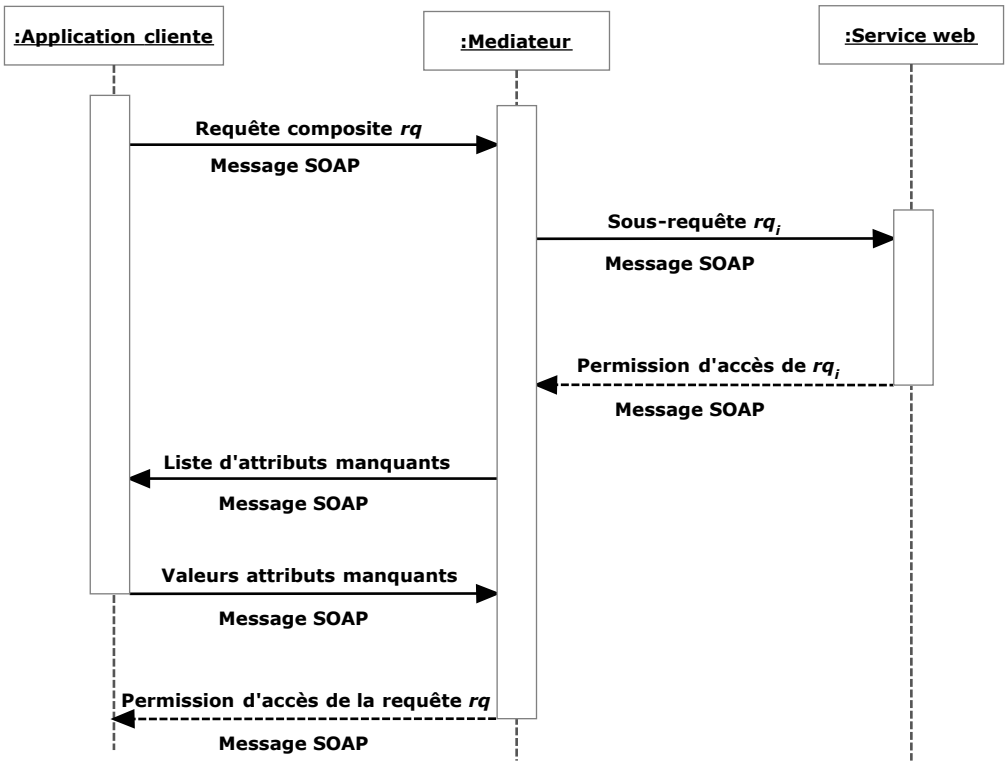

FIGURE 5.2 – Diagramme de séquence

5.5 Description des différents composants du système

Dans cette section, nous présentons les différents acteurs du système.

5.5.1 Application cliente

L'application cliente, représente l'émetteur de la requête (rq) dans le système, elle est implémentée en JAVA fait appel aux médiateurs, et fournit une interface qui permet de tester le prototype, en envoyant une requête et en recevant la permission d'accès de la requête.

5.5.2 Médiateur

Le médiateur traite la requête émise par l'application cliente, il est dynamique et sélectionné aléatoirement parmi plusieurs services Web afin d'éviter les attaques des uti-

lisateurs malveillants. Les différents modules du médiateur sont représentés si dessus.

5.5.2.1 Bibliothèque CoGITaNT

La bibliothèque CoGITaNT fournit les outils nécessaires à la création et à la manipulation des graphes conceptuels.

5.5.2.2 Ontologie de domaine

L'ontologie de domaine au niveau du médiateur fournit le vocabulaire utilisé par les différents services Web afin de formuler les permissions d'accès. Elle est utilisée afin d'adapter les permissions d'accès issues des services Web qui n'utilisent pas notre politique de contrôle d'accès. L'ontologie de domaine est créée avec l'outil TooCom, la figure 5.2 illustre un exemple d'ontologie de domaine du médiateur réalisée avec l'outil TooCom.

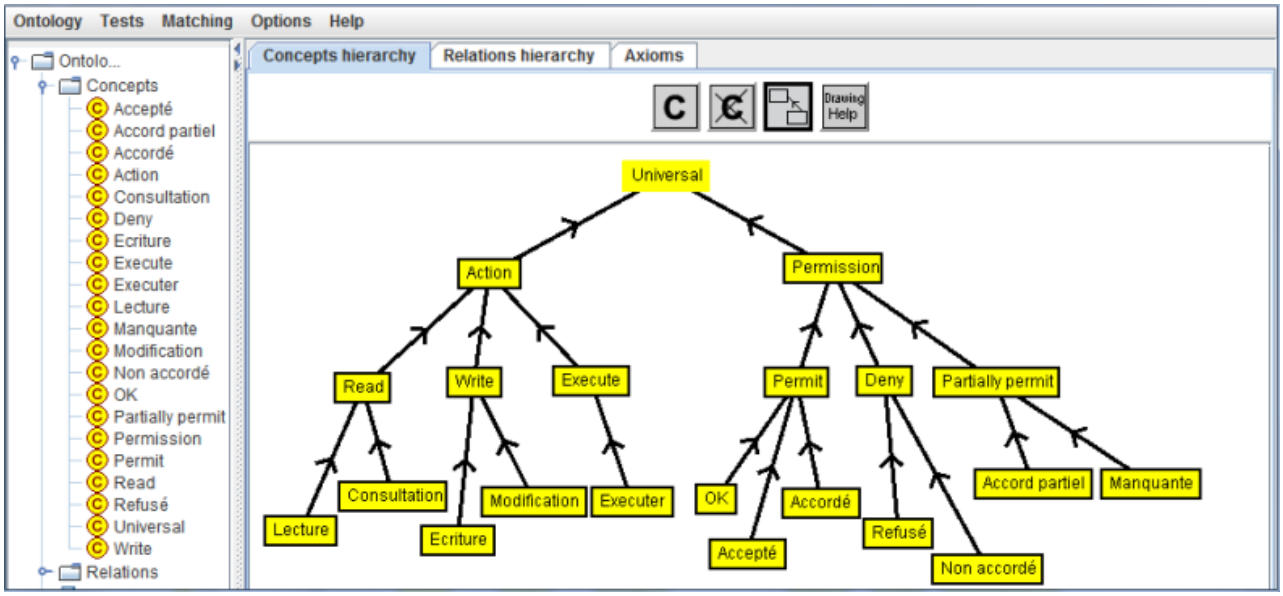

FIGURE 5.3 – Ontologie du domaine du médiateur

5.5.2.3 Parser GC

Le rôle du Parser GC est la translation de la requête émise par l'application cliente en graphe conceptuel, pour qu'elle soit traitée par le module contrôle d'accès, et la translation des permissions d'accès issues des services Web appartenant aux cercles de confiances qui n'utilisent pas notre politique de contrôle d'accès. Le Parser GC utilise la bibliothèque CoGITaNT pour construire les graphes conceptuels.

Module communication Le module communication reçoit les différents messages SOAP issus de l'application cliente et des services Web des différents cercles de confiances, les transmettant ainsi au module contrôle d'accès, il transmet aussi les requêtes (rq_i) issues

du module contrôle d'accès aux différents services Web concernés et la permission d'accès à la requête (rq) sous forme d'un message SOAP.

5.5.2.4 Module contrôle d'accès

Le module contrôle représente l'implémentation de modèle du contrôle d'accès, il utilise la bibliothèque CoGITaNT et l'ontologie de domaine. Il communique avec le module communication qui lui fournit une interface de communication. Les différentes étapes du module contrôle d'accès sont :

- Décomposition de la requête (rq) en sous-requêtes (rq_i) ;
- Transmission des sous-requêtes (rq_i) au module communication afin que ce dernier les transmettent aux différents services Web simples sous forme d'un message SOAP ;
- Réception des permissions d'accès des différentes requêtes (rq_i) ;
- Adaptation des permissions d'accès issues des services Web qui n'utilisent pas notre politique de contrôle d'accès en utilisant l'ontologie du domaine ;
- Résolution du conflit ;
- Négociation avec le client en utilisant le module communication ;
- Envois de la permission d'accès de la requête (rq) à l'application cliente.

5.5.3 Service Web

Les services Web sont regroupés dans des cercles de confiance selon la politique de contrôle d'accès. Le cercle de confiance A regroupe les services Web qui utilisent la politique de contrôle d'accès du modèle proposé.

5.5.3.1 Bibliothèque CoGITaNT

Elle fournit les mêmes fonctionnalités que la Bibliothèque CoGITaNT du médiateur.

5.5.3.2 Ontologie de domaine

L'ontologie de domaine au niveau du cercle de confiance fournit le vocabulaire utilisé par les différents services Web du cercle de confiance afin représenter les différents attributs utilisés pour l'attribution du profil, les opérations et les paramètres utilisés. L'ontologie de domaine du cercle de confiance est créée avec l'outil TooCom.

5.5.3.3 Historique

Historique est une base de connaissances qui contient l'historique des utilisateurs qui ont déjà utilisé le service Web.

5.5.3.4 Module Accès

Le module accès représente l'implémentation de la politique de contrôle d'accès du modèle proposé. Ce module est implémenté en C++ en utilisant la bibliothèque CoGITaNT, les différentes étapes du module accès sont les suivantes :

- Construction du profil utilisateur, cette étape est réalisée en utilisant l'historique du sujet et l'ontologie du domaine du cercle de confiance ;
- Capture du contexte utilisateur nécessaire à la vérification des permissions d'accès ;
- Attribution du rôle à partir du profil utilisateur ;
- Vérification des permissions d'accès ;
- Envoi des permissions d'accès via un message SOAP.

5.6 Scénarios d'exécution

Nous illustrons dans cette section quelques scénarios d'exécutions.

5.6.1 Présentation de l'UDDI étendu

L'UDDI étendu contient 10 services Web appartenant à 3 cercles de confiance différents telle que le cercle de confiance A qui regroupe les services Web qui utilisent la politique de contrôle d'accès du modèle proposé, les cercle de confiance B et C utilisent des politiques de contrôle d'accès différents. Les différents services Web utilisés sont :

- **WS_Bank :** C'est un service Web d'une banque qui fournit des services de consultation (Consultation), de virement (Transfer), et de retrait (Withdrawal)à ses clients ;
- **WS_Hotel :** C'est un service Web d'un Hôtel qui permet aux touristes de consulter leur brochure (Brochure), faire des réservations (Reservation) et la confirmation (Confirmation) de leurs réservations ;
- **WS_Excursion :** C'est un service Web d'une agence touristique qui donne la possibilité aux touristes de consulter (Consult) les différentes excursions proposées et de réserver (Book) directement via le service Web ;
- **WS_Bookstore :** C'est un service Web d'une librairie qui propose aux utilisateurs du service la possibilité de chercher (Search), commander des livres (Booking) ;
- **WS_University :** C'est un service Web d'une université, qui donne la possibilité aux différents utilisateurs de consulter la liste des formations proposées (Find_training), donner des informations sur les enseignants (Find_teacher) et de faire des inscriptions (Enrollment) ;
- **WS_Weather :** C'est un service Web qui donne les prévisions météo (Weather_forecasting) et les prévisions des plages (Forecast_beaches) ;
- **WS_Travel :** C'est un service Web d'une agence de voyage qui permet de consulter les différentes destinations de voyage (Consultation), de réserver des voyages via ce

service Web (Reservation) ;

- **WS_NeWS :** C'est un service Web d'information qui donne les informations dans le monde (Find_news) et offre la possibilité de télécharger des journaux (Get_news) ;
- **WS_leasing :** C'est un service Web d'une agence de location de véhicules qui permet aux utilisateurs de louer (Rent) des véhicules via ce service Web ;
- **WS_Article :** C'est un service Web qui regroupe des articles scientifiques publiés et donne la possibilité de chercher un article (Find_article), de consulter un article (Get_article), et consulter la liste des articles (Article_list).

La tableau 5.1 représente les différents services Web, leurs opérations et les cercles de confiances où ils appartiennent.

La figure 5.4 illustre la représentation en graphe conceptuel du service Web **WS_Excursion**.

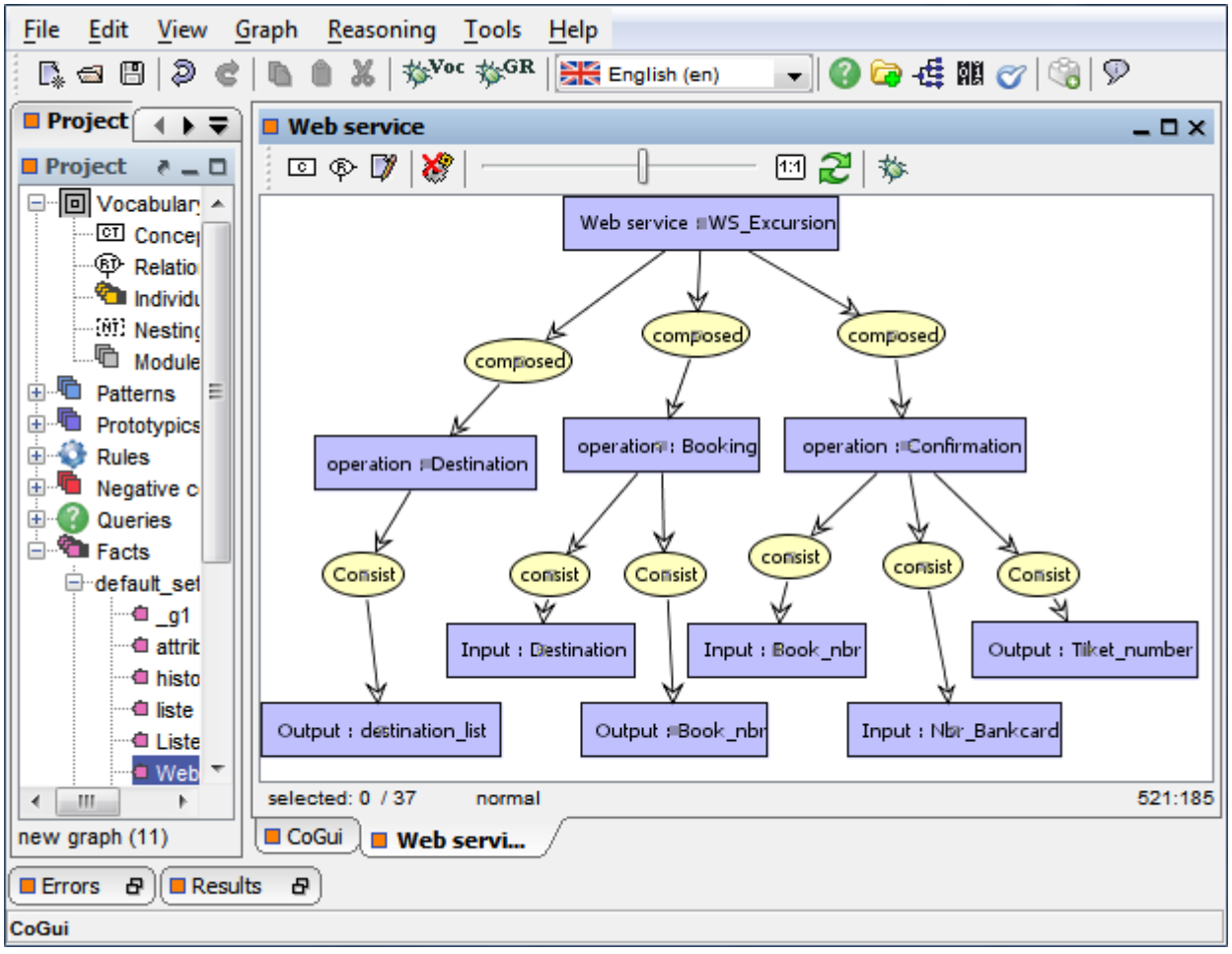

FIGURE 5.4 – Service Web SW_Excurtion

5.6.2 Cas 1 : Requête composite vers le cercle de confiance A

Le cas 1 représente un scénario d'exécution d'une requête composite vers le cercle de confiance A.

Service Web	**Opération**	**Entrée**	**Sortie**	**Cercle de confiance**
WS_Bank	Withdrawal	Amount	Balance	Cercle de confiance A
	Transfer	Compt_Nbr	Transaction_Nbr	
		Amount		
	Consultation		Balance	
WS_Hotel	Consulter		Room_List	
	Reservation	Room_Nbr	Reservation_Nbr	
	Confirmation	Room_Nbr	Confirmation_nbr	
		BankCard_Nbr		
WS_Excursion	Destination		Destination_List	
	Reservation	Book	Book_Nbr	
	Confirmation	Destination	Ticket_Nbr	
		BnkCard_Nbr		
WS_University	Find_Training	key_Word	Training	
	Find_ Teacher	Teacher_Name	Teacher	
	Enrollment	Matricul	Form	
WS_Travel	Find	Destination	Travel_List	Cercle de confiance B
	Reservation	Destination	Reservation_Nbr	
	Confirmation	Reservation_Nbr	Confirmation_Nbr	
		Credit_Card		
WS_Bookstor	Search	Book_Title	Book	
	Booking	Book_Title	Num_Commande	
		Card_Nbr		
	Book_List	category	List	
WS_Weather	weather_ Forecasting	Place	Forecasting	
	forecast_ Beaches	Place	Forecast	
WS_leasing	Car_List		Cars_List	Cercle de Confiance C
	Rent	Car_Nbr	Confirmation_Nbr	
		Lasted		
		Card_Nbr		
WS_News	Find_NeWS	Country	NeWS	
	Get_NeWS	Name_ Journal	Journal	
WS_Article	Find_Article	Article	Result	
	Get_NeWS	Article	Result	
	Article_List		Result	

TABLE 5.1 – Description des services Web

5.6.2.1 La requête

Soit la requête ($rq1$) suivante :

rq1 = (WS_Excursion.Reservation("Alpes"), Execute) + (WS_Hotel.Reservation(24), Execute).

La requête ($rq1$) représente une requête composite vers les deux services Web (WS_Excursion) et (WS_Hotel) où les deux services Web appartiennent au même cercle de confiance A. La requête ($rq1$) signifie que le sujet veut faire la réservation d'une excursion aux (Alpes) au niveau du service Web (WS_Excursion) ou faire un réservation de la chambre d'hôtel (24) au niveau du service Web (WS_Hotel), le (ou) dans la requête est représenté par l'opérateur de choix (+).

La figure 5.5 représente l'interface de l'application qui permet au sujet de saisir sa requête.

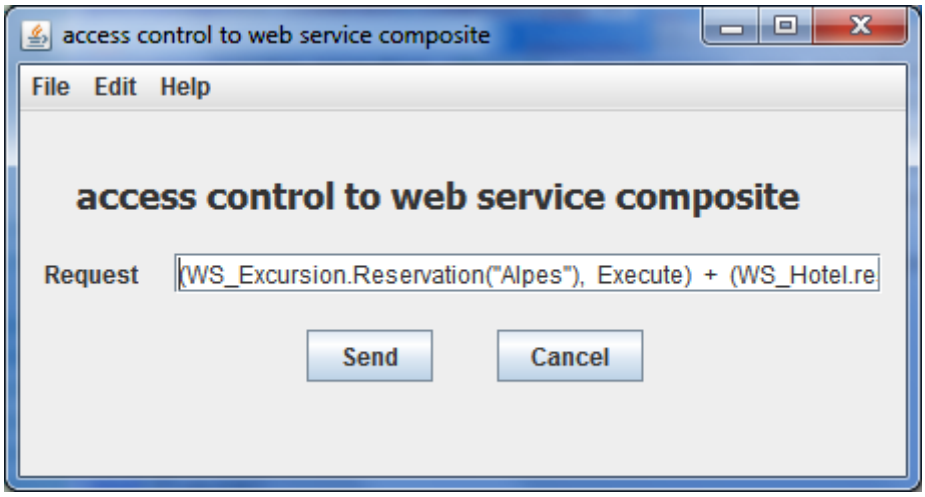

FIGURE 5.5 – Envoi de la requête

5.6.2.2 Le sujet

Le sujet (s) est l'utilisateur de l'application cliente, il est représenté comme suite :
s = (Name, Yanis),(First name, Mohamed), (Level, Bac+5), (Profession, Programmer), (Workplace, MicroSoft), (account, 23652365)

La figure 5.6 illustre la représentation du sujet (s) sous forme d'un graphe conceptuel.

5.6.2.3 Représentation de la requête en graphe conceptuel

Une fois la requête reçue par le médiateur, la première étape consiste à traduire la requête en un graphe conceptuel, cette étape est réalisée par le Parser CG en utilisant

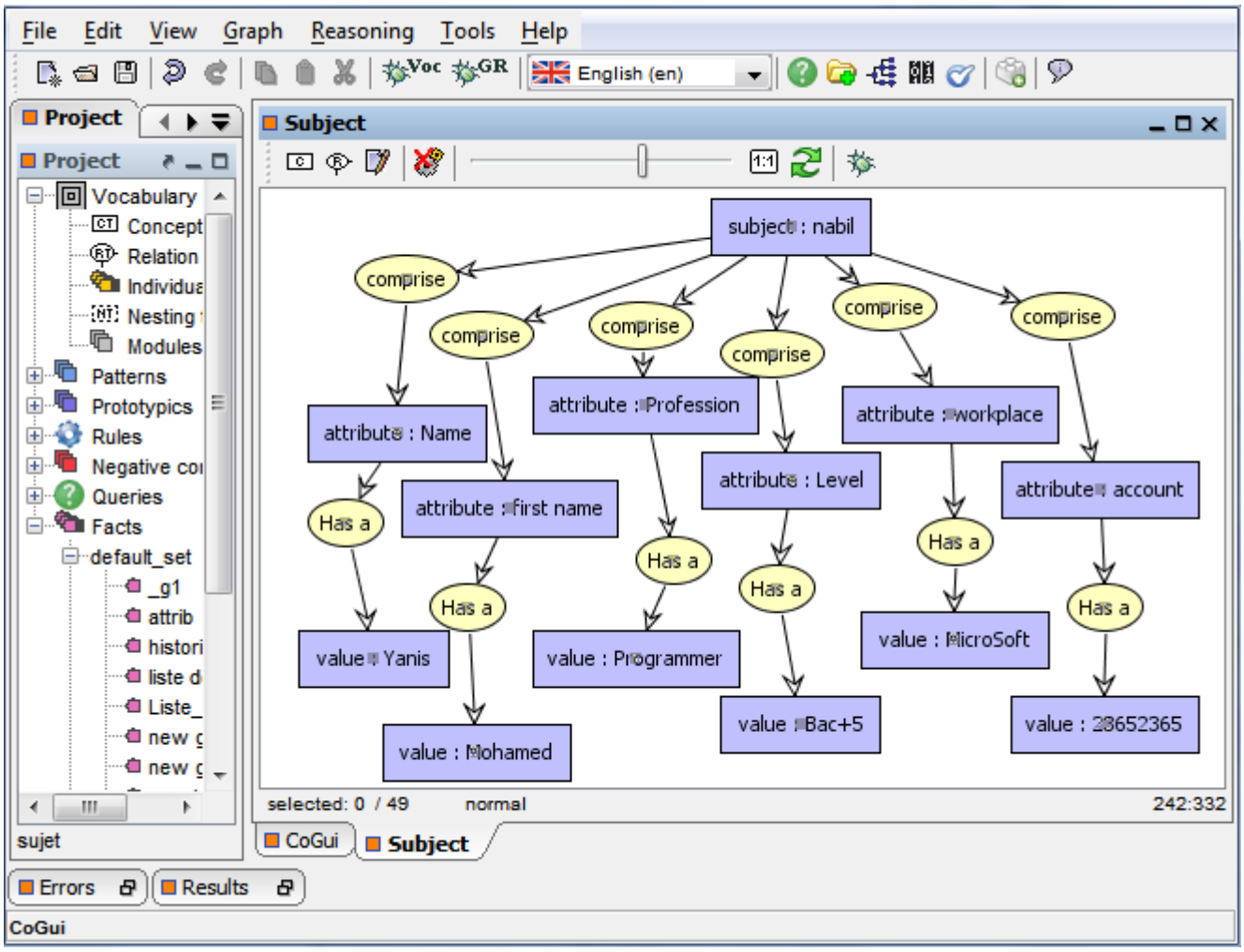

FIGURE 5.6 – Représentation du sujet s

la bibliothèque CoGITaNT, dans la bibliothèque CoGITaNT, un graphe conceptuel **CoGITaNT : : Graph** [10] est représenté sous la forme d'un ensemble de n øeuds **CoGITaNT : : GraphObject** et d'un ensemble d'arêtes **CoGITaNT : : Edge**. Chaque objet d'un graphe est identifié de façon unique par le **CoGITaNT : : iSet** qui repère cet objet dans l'ensemble des objets. La figure 5.7 représente le graphe conceptuel de la requête $rq1$.

5.6.2.4 Décomposition de la requête

L'étape de décomposition de la requête consiste à décomposer le graphe de la requête($rq1$) en sous-requête ($rq1_i$), chaque sous-requête ($rq1_i$) est ensuite envoyée au service Web simple concerné pour la vérification des permissions d'accès de ces dernières. La figure 5.8 illustre la requête simple destinée au service Web (WS_Excursion) Le tableau 5.2 représente le résultat de la décomposition de la requête ($rq1$).

10. Représente la classe graphe conceptuel

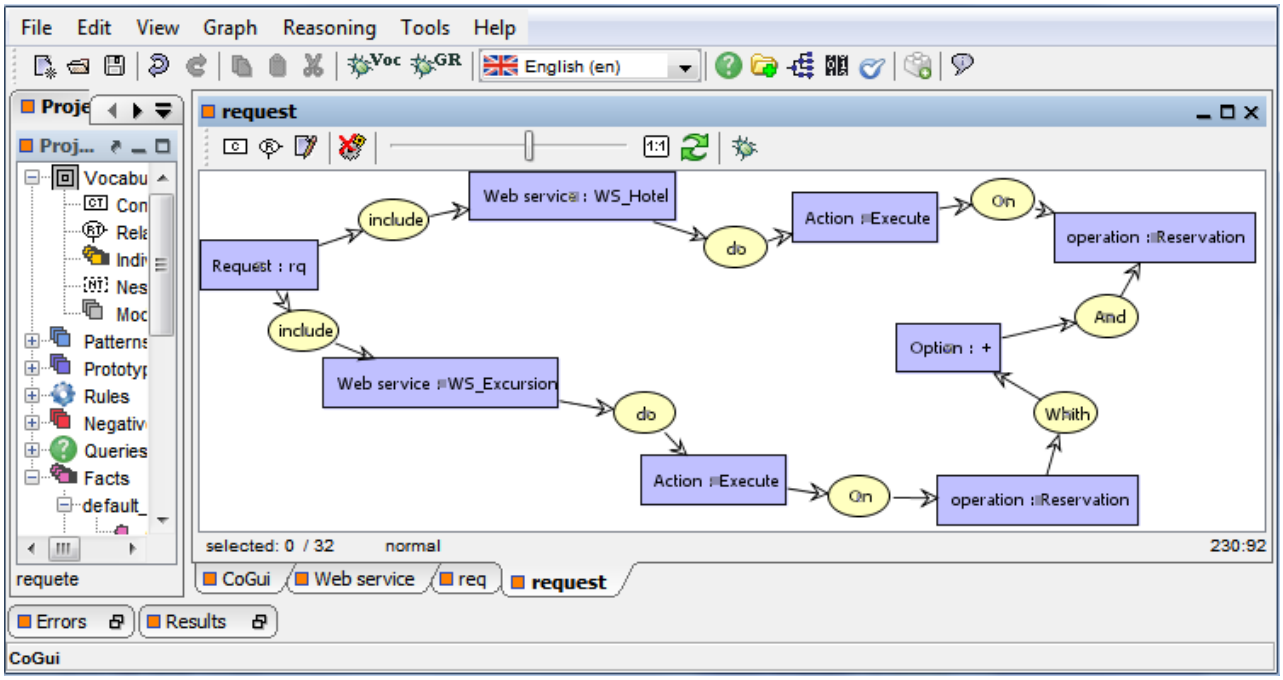

FIGURE 5.7 – Représentation de la requête *rq1*

Remarque : les étapes de construction du profil, capture du contexte utilisateur, attribution du rôle et vérification des permissions d'accès se déroulent au niveau des services Web du cercle de confiance A en parallèle. À l'exception de ces étapes, toutes les autres étapes se déroulent au niveau du médiateur.

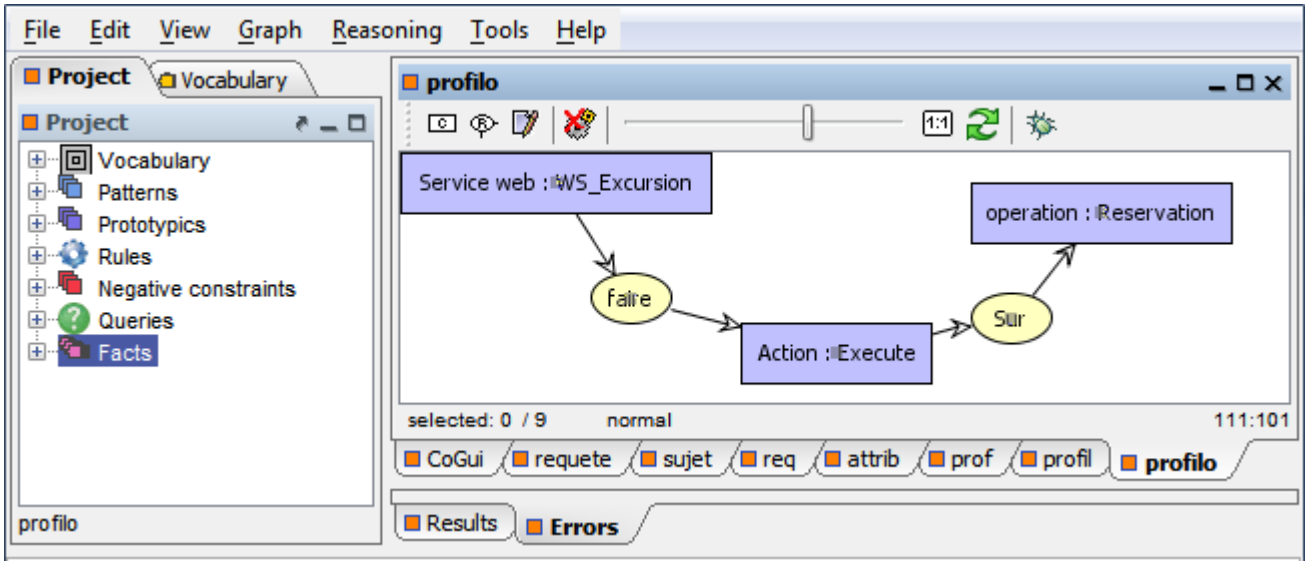

FIGURE 5.8 – Représentation de la sous-requête $rq1_i$

5.6.2.5 Construction du profil

Une fois que le service Web a reçu la requête ($rq1_i$), la première étape effectuée par le module accès est la construction du profil utilisateur, à partir des attributs et l'his-

Sous-requête	Service Web	Opération	Action
$rq1_1$	WS_Excursion	Reservation	Execute
$rq1_2$	WS_Hotel	Reservation	Execute

TABLE 5.2 – Liste de sous-requêtes $rq1_i$

torique du sujet (s), le module accès créé un profil temporaire en utilisant les classe **CoGITaNT : :OpeDisjointSum** et **CoGITaNT : :Environment : :normalize** qui permettent respectivement de faire la somme disjointe et la normalisation d'un graphe. Ensuite, en projetant ce profil temporaire sur la liste des profils du service Web en utilisant la classe **CoGITaNT : :OpeProjection**, il sélectionne le profil utilisateur qui correspond aux sujet (s). La figure 5.9 illustre le graphe conceptuel du profil attribué au sujet (s). Dans ce profil, la dimension historique est composée des attributs (connections_nbr et reservation_nbr) qui représentent respectivement le nombre de fois que le sujet a visité le service Web et le nombre de réservations que le sujet a effectué.

5.6.2.6 Capture du contexte utilisateur

Le fournisseur du service Web capture les contextes nécessaires à la vérification des permissions d'accès. Dans ce scénario, en utilisant l'adresse IP [11] du sujet, le fournisseur du service Web (WS_Excursion) capture le contexte (current_place) qui représente le lieu d'où le sujet (s) a émis sa requête.

5.6.2.7 Attribution du rôle

L'attribution du rôle se fait par le profil du sujet. À partir de ce profil, un graphe conceptuel (Pfr)(voir la figure 5.10) est construit, ce graphe contient un concept générique de type rôle et le concept profil du sujet, ce dernier sera projeté sur le graphe de la liste des rôles en utilisant la classe **CoGITaNT : : OpeProjection**, afin de sélectionner le rôle du sujet en utilisant la classe **CoGITaNT : : ResultOpeProjection** qui fournit le résultat de la projection. la figure 5.11 illustre le résultat de l'opération de projection.

5.6.2.8 Vérification des permissions d'accès

Les permissions d'accès dans le module accès sont organisées sous forme de règles de graphes conceptuels. Une règle est représentée par la classe **CoGITaNT : : Rule** qui est composée de deux graphes, l'hypothèse et la conclusion. La méthode **CoGITaNT : : Rule : :hypothesis()** permet d'accéder au graphe hypothèse, et **CoGITaNT : : Rule : : conclusion()** au graphe conclusion.
Chaque rôle contient un ensemble de règles, chaque règle représente une contrainte sur

11. Internet Protocole

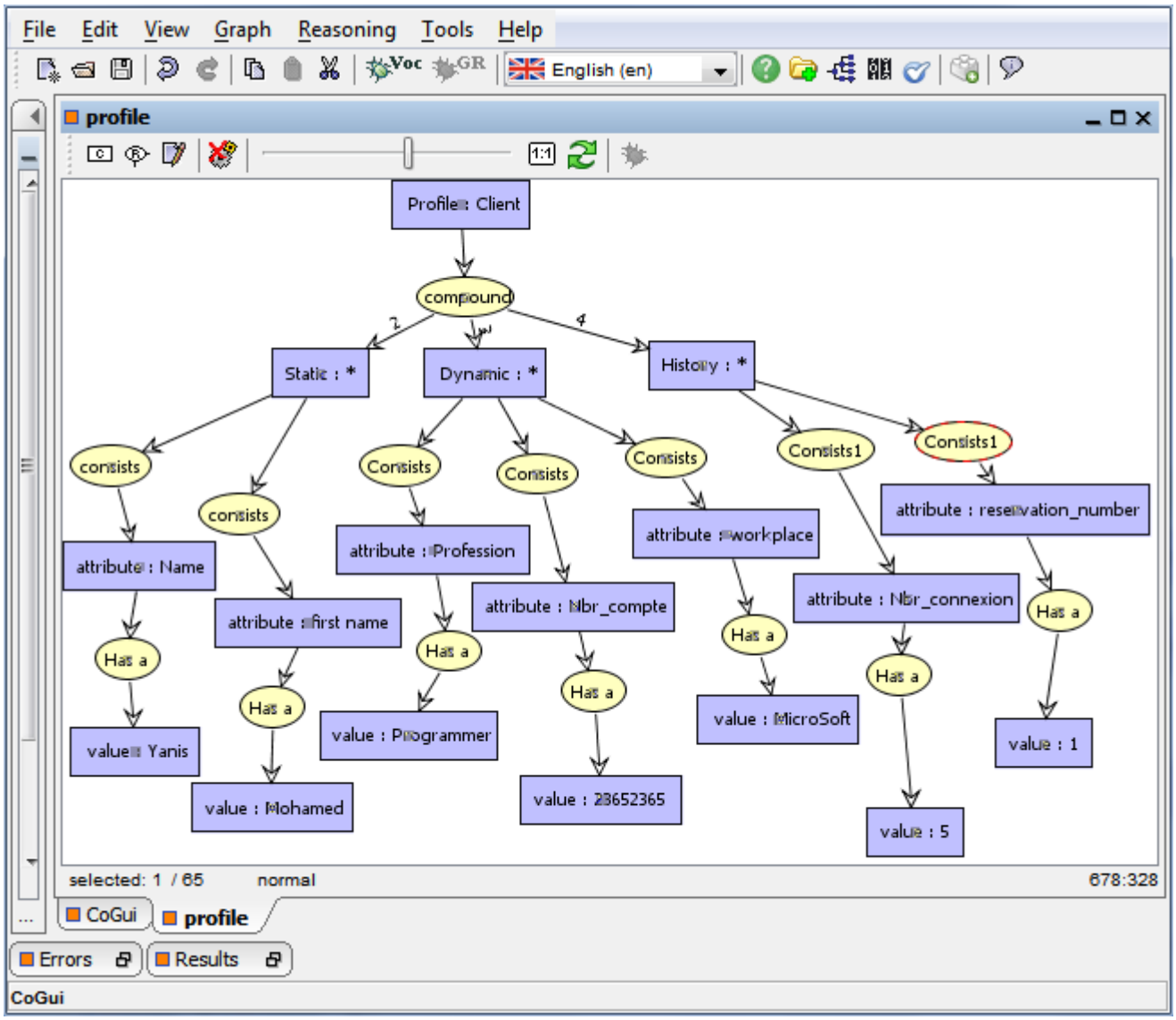

FIGURE 5.9 – Représentation du profil du sujet s

FIGURE 5.10 – Graphe conceptuel Prf

la permission d'accès. la figure 5.12 illustre la règle utilisée pour vérifier la permission d'accès de la sous-requête (rq_2).

Une fois la permission d'accès sélectionnée, le service Web envoie la permission d'accès au médiateur via un message SOAP.

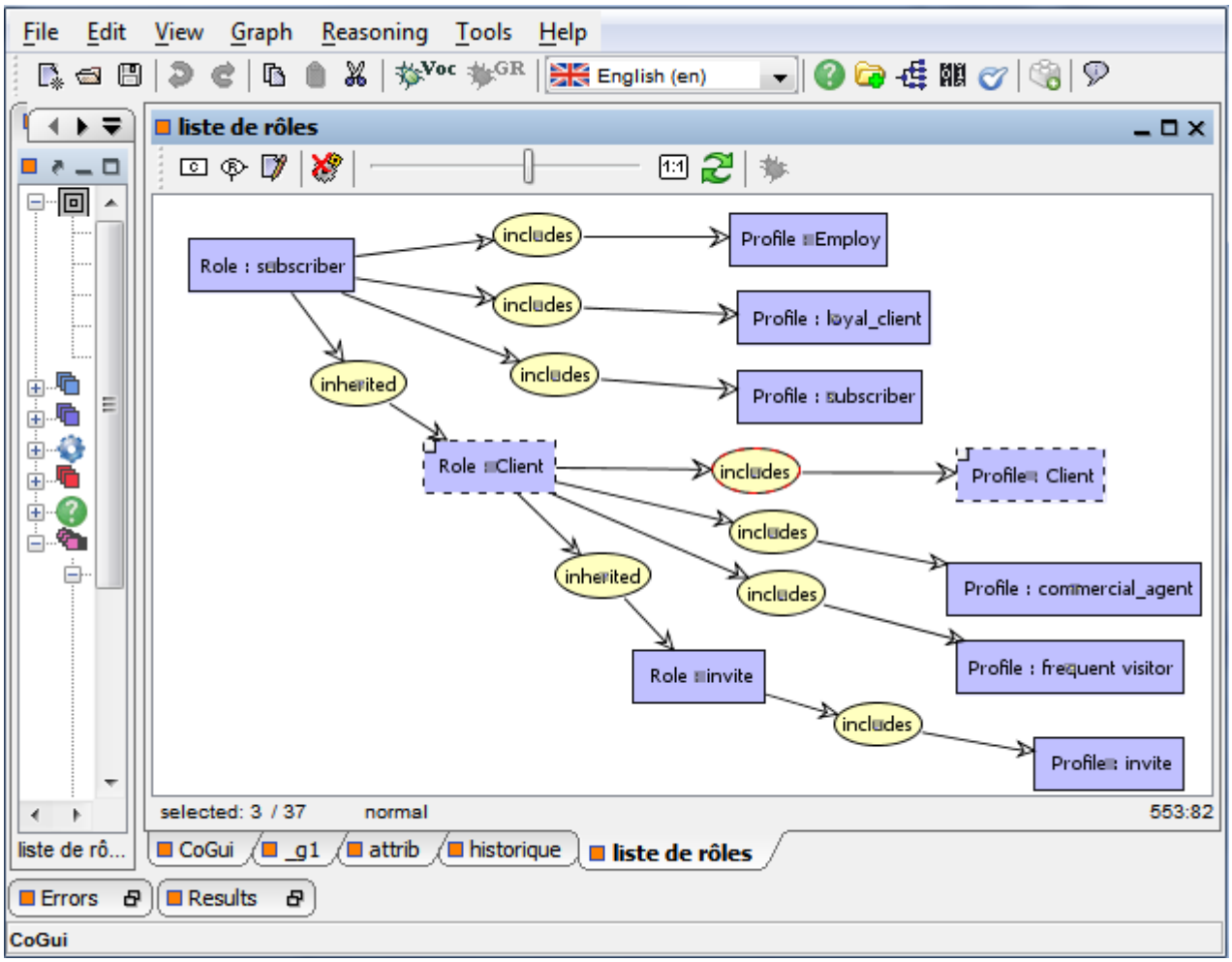

FIGURE 5.11 – Résultat de la projection du rôle

5.6.2.9 Réception des permissions d'accès

Le médiateur définit une période de temps pendant laquelle il attend les permissions d'accès des sous-requêtes ($rq1_i$) issues des différents services Web. Si à la fin de cette période une permission d'accès n'est pas reçue, alors cette dernière sera considérée comme une permission refusée (*deny*).

Le tableau 5.3 illustre les permissions d'accès aux requêtes ($rq1_i$) issues des différents services Web impliqués dans la requête ($rq1$). Dans ce tableau, la permission de la sous-requête ($rq1_1$) est acceptée(*permit*), et la permission à la sous-requête ($rq1_2$) est refusée (*deny*).

Sous-requête	**Service Web simple**	**permission d'accès**
$rq1_1$	WS_Excursion	*permit*
$rq1_2$	WS_Hotel	*deny*

TABLE 5.3 – Permissions d'accès des sous-requêtes rq_i

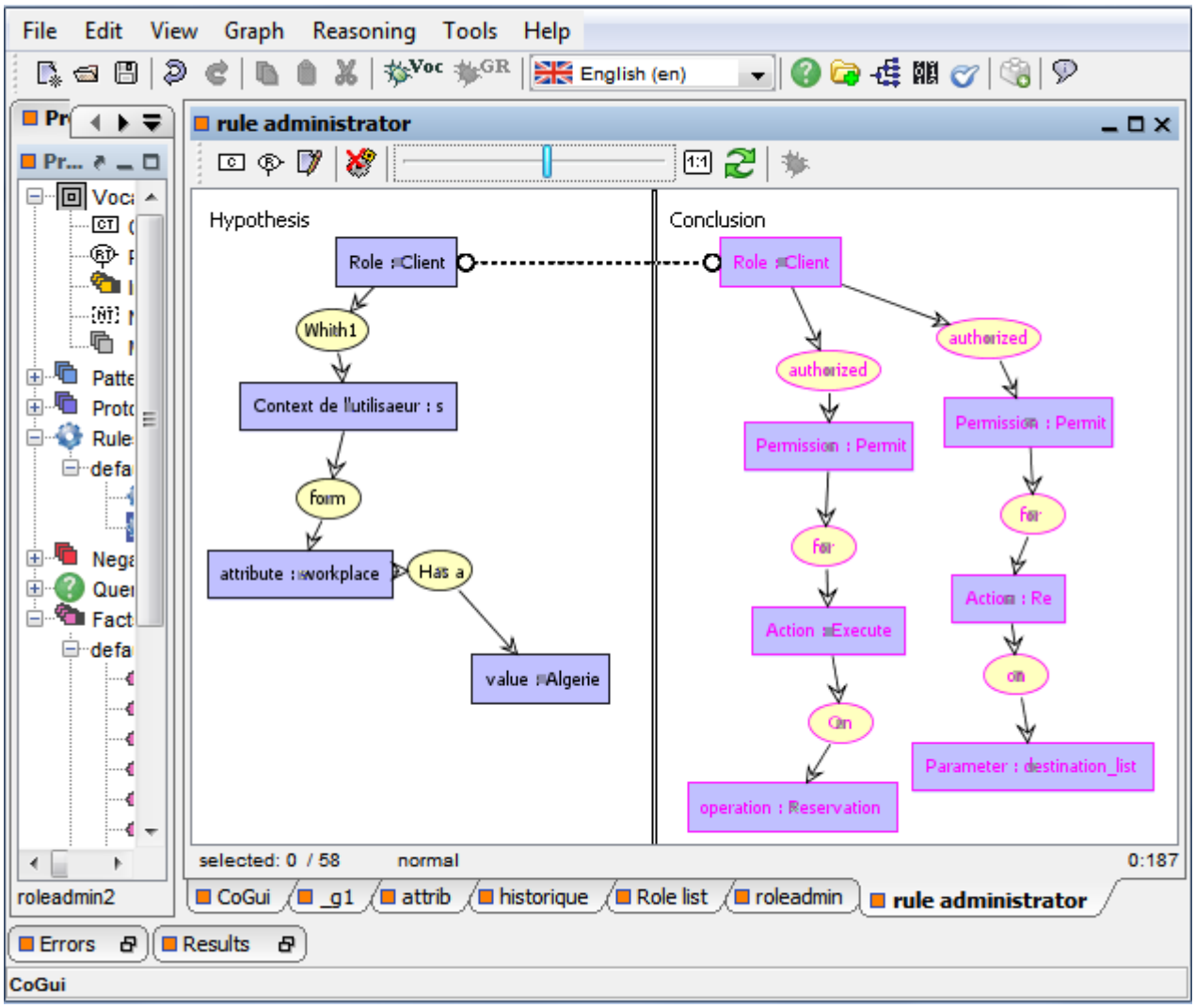

FIGURE 5.12 – Règle de graphe

5.6.2.10 Résolution du conflit

La résolution du conflit intervient dans le cas où des permissions d'accès positive (*permit*) et négative (*deny*) sont présentent au même temps dans la requête composite ($rq1$), cette étape fournie la décision de lancer ou de ne pas lancer la négociation avec le sujet et une permission d'accès à la requête composite de départ.
Dans le scénario présenté, la résolution de conflit est nécessaire pour fournir la permission d'accès à la requête ($rq1$). En effet, la requête ($rq1$) (voir la figure 5.13) contient une permission positive(*permit*), et une permission négative (*deny*).

Construction de la fonction logique A partir du graphe de la figure 5.13, on construit une fonction logique (f) tel que :

- Les permissions d'accès positives (*permit*) auront la valeur (1) ;
- Les permissions d'accès négative (*deny*) auront la valeur (0) ;
- L'option de composition au choix (+) sera représentée par l'opérateur logique OU ($\vee$) ;

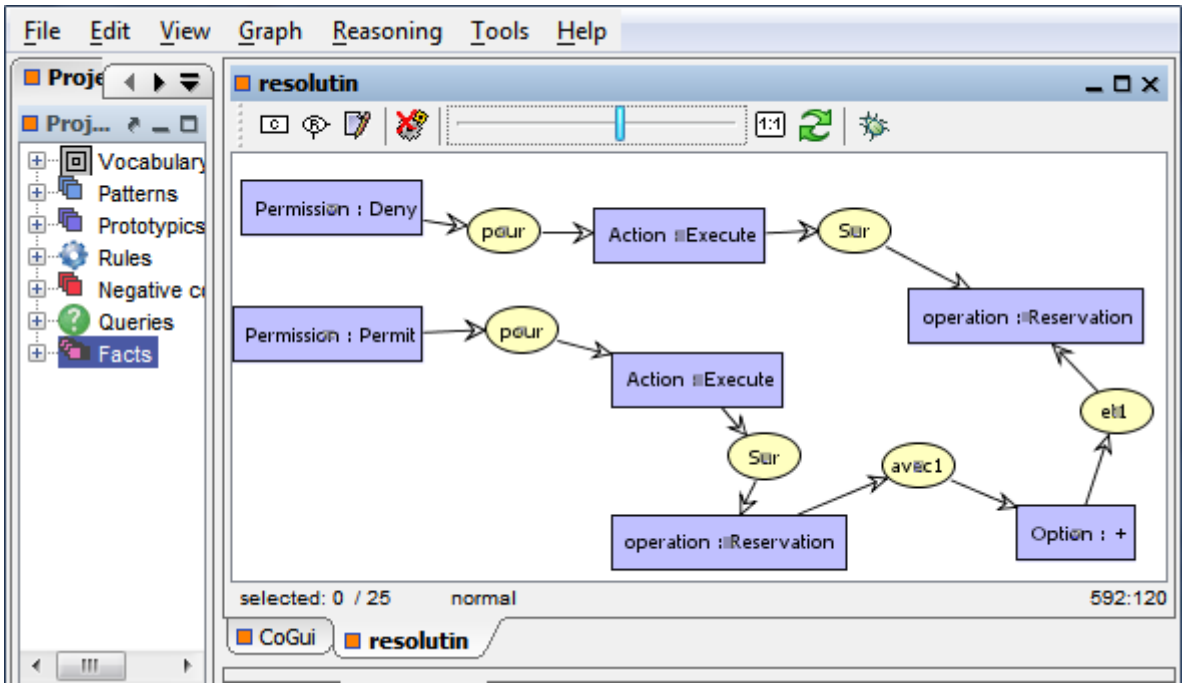

FIGURE 5.13 – Composition des permissions d'accès

– L'option de composition en série (-) sera représentée par l'opérateur logique ET ($\wedge$) ;

La fonction f du scénario est comme suit :
$f = 1 \vee 0$; le résultat de la fonction (f) est (1), ce qui implique que la permission d'accès de la requête ($rq1$) est positive ($permit$).

5.6.3 Cas 2 : Requête composite avec négociation

Le cas 2 représente un scénario d'exécution d'une requête composée de deux servies Web du cercle de confiance A avec une négociation à la fin.

5.6.3.1 La requête

Soit la requête ($rq2$) suivante :

rq2 = (WS_University.Enrollement(inf02098), Execute) - (WS_Bank.Withdrawal (254302154, 500), Execute)

La requête ($rq2$) représente une requête composite vers les deux services Web (WS_University) et (WS_Bank) où les deux services Web appartiennent au même cercle de confiance A. La requête ($rq2$) signifie que le sujet veut faire une inscription au niveau du service Web (WS_University), (inf02098) représente la matricule du sujet et faire un virement bancaire au niveau du service Web (WS_Bank) où (254302154) représente le numéro de compte et (500) représente le montant à virer, le (ET) est représenté par le (-) dans la requête.

5.6.3.2 Décomposition de la requête

Le tableau 5.7 représente le résultat de la décomposition de la requête ($rq2$).

Sous-requête	**Service Web simple**	**Opération**	**Action**
$rq2_1$	WS_Univertity	Enrollement	Execute
$rq2_2$	WS_Bank	Withdrawal	Execute

TABLE 5.4 – Liste des sous-requêtes $rq2_i$

5.6.3.3 Réception des permissions d'accès

Le tableau 5.5 représente les permissions d'accès des sous-requêtes ($rq2_1$) et ($rq2_2$)

Sous-requête	**Service Web simple**	**permission d'accès**	**Attribut manquant**
$rq2_1$	WS_University	*permit*	
$rq2_2$	WS_Bank	$p - permit$	Bank_Card

TABLE 5.5 – Permissions d'accès des requêtes $rq2_i$

Dans le tableau 5.5, la permission d'accès fournie par le service Web (WS_Bank) est partiellement acceptée ($p-permit$) avec un attribut manquant (Bank_Card) qui représente le numéro de la carte bancaire du sujet. Afin que cette permission soit positive (*permit*), le sujet doit fournir le numéro de sa carte bancaire via une négociation. Mais avant de lancer la négociation, le médiateur doit vérifier que l'attribut carte bancaire doit influencer sur la permission finale de la requête composite ($rq2$).

5.6.3.4 Résolution du conflit

La fonction logique (f) de la requête ($rq2$) est comme suit :

$f = 1 \wedge x$, tel que (x) représente la permission partiellement satisfaite ($p - permit$).

Le tableau 5.6 illustre les deux principales vérifications de l'algorithme de résolution du conflit (section 4.6.2.6) sur la fonction (f), la première vérification consiste à vérifier l'importance des attributs manquants, sachant que remplacer ($p - permit$) par (*permit*) signifie que le sujet va fournir les attributs manquants, la valeur de (f) est 1 ce qui implique que la permission de la requête ($rq2$) est (*permit*),(remarque : si la valeur de (f) était 0 alors la permission de ($rq2$) est (*deny*) et fin de l'algorithme). La deuxième vérification consiste à vérifier la nécessité de lancer la négociation avec le sujet, sachant que remplacer ($p - permit$) par (*deny*) signifie que le sujet ne va pas fournir les attributs

manquants, la valeur de (f) est 0 ce qui implique que la permission de $(rq2)$ est $(deny)$, alors lancer la négociation avec le sujet (remarque : si la valeur de (f) était 1 alors la négociation n'est pas nécessaire la permission est $(permit)$ et fin de l'algorithme).

Vérification	**Description**	**Valeur de x**	**Valeur de f**
1	Remplacer chaque permission $(p - permit)$ par $(permit)$	1	1
2	Remplacer chaque permission $(p - permit)$ par $(deny)$	0	0

TABLE 5.6 – Permissions d'accès des requêtes rq_i

5.6.3.5 Négociation avec le sujet

Le processus de négociation consiste à envoyer la liste des attributs manquants au sujet, et attendre la réponse du sujet. Si à la fin de sa session, le sujet ne fournit pas les attributs manquants alors la permission d'accès est $(deny)$. Dans le scénario du cas 2, le médiateur va demander au sujet de fournir l'attribut (Bank_Card) sous forme d'un message SOAP (voir la figure 5.16). Le sujet va fournir le numéro de sa carte bancaire en saisissant ce dernier sur l'application client (voir la figure 5.14). Enfin, le médiateur envoie la permission d'accès final de la requête composite $(rq2)$ qui est $(permit)$ au sujet (voir la figure 5.15).

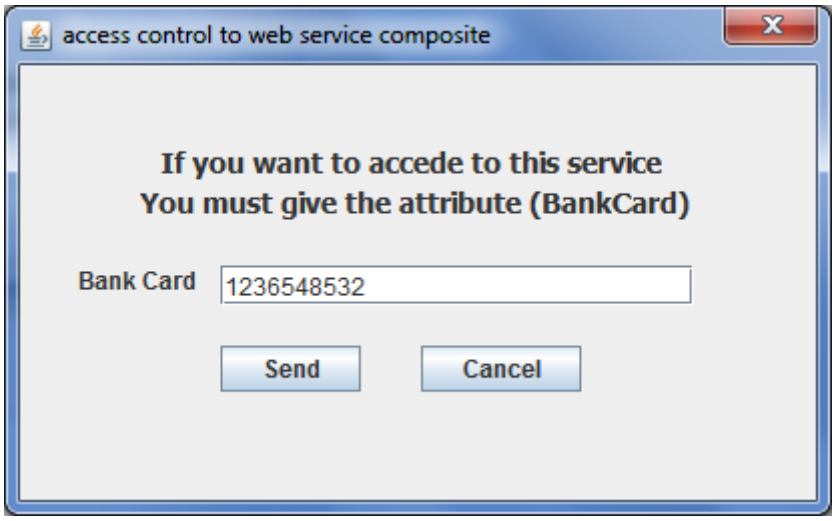

FIGURE 5.14 – Ajout des attributs manquants

5.6.4 Cas 3 : Requête composite vers plusieurs cercles de confiance

Le cas 3 représente un scénario d'exécution d'une requête composée de trois servies Web de cercles de confiances différents. Avec adaptation des permissions d'accès, c'est le cas le plus complexe.

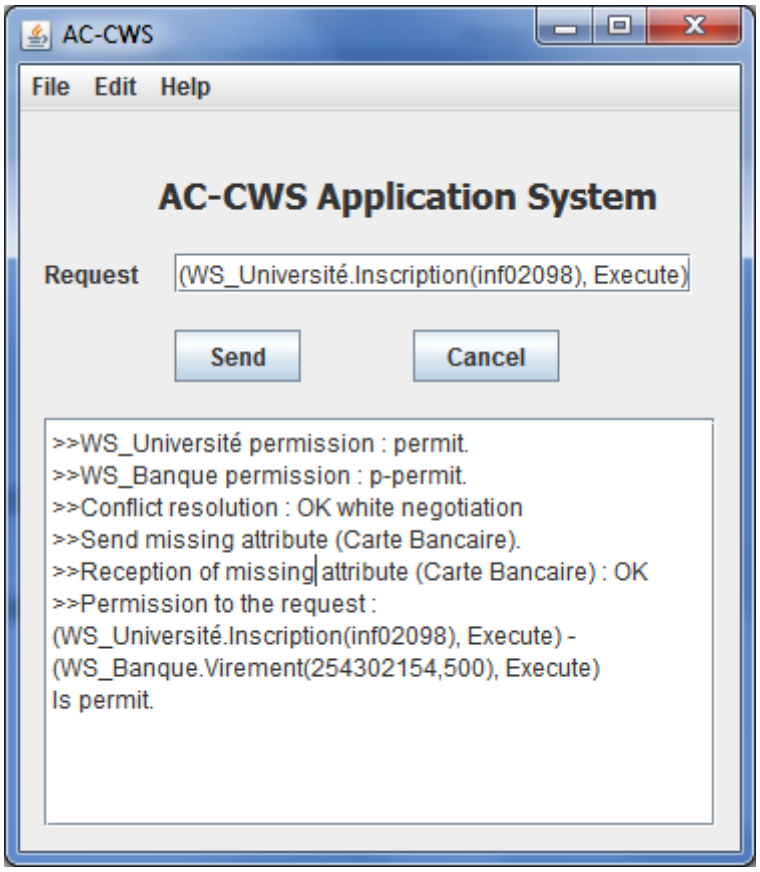

FIGURE 5.15 – Permission de la requête ($rq2$)

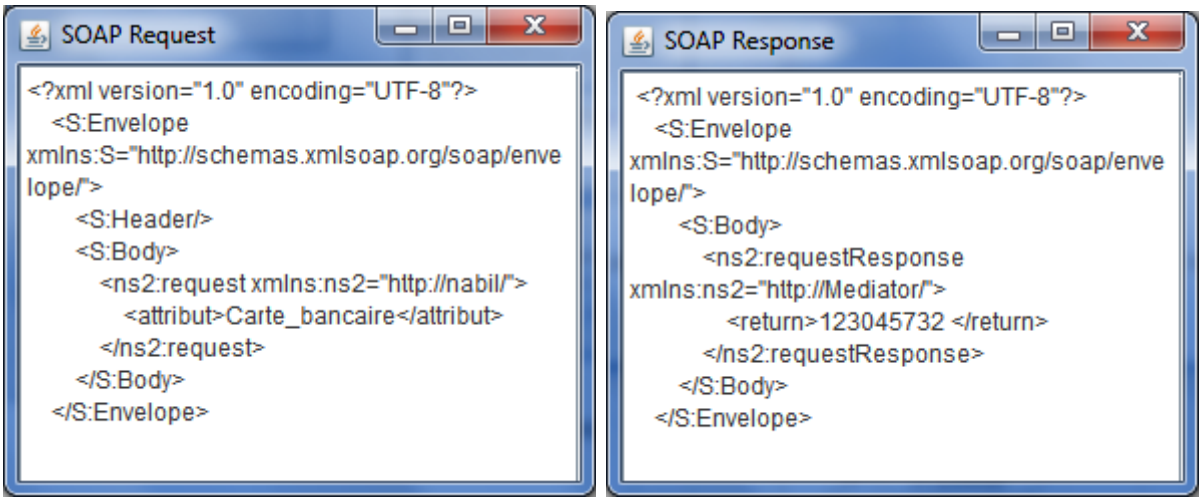

FIGURE 5.16 – Messages SOAP

5.6.4.1 La requête

Soit la requête ($rq3$) suivante :

rq3 = (WS_Travel.Reservation(France), execute) - (WS_Hotel.Room_list, read) - (WS_Leasing.Car_list, write).

La requête ($rq3$) représente une requête composite vers trois services Web (WS_Travel), (WS_Hotel) et (WS_Leasing) issus de trois cercles de confiances différents. La requête ($rq3$) signifie que le sujet veut faire une réservation de voyage en France, et consulter la liste des chambres dans le service Web (WS_Hotel) et modifier la liste des voitures.

5.6.4.2 Décomposition de la requête

Le tableau 5.7 représente les sous-requêtes ($rq3_i$) après la décomposition de la requête ($rq3$).

Sous-requête	**Service Web**	**Opération**	**Action**	**Cercle de confiance**
$rq3_1$	WS_Hotel	Room_list	Read	cercle de confiance A
$rq3_2$	WS_Voyage	Reservation	Execute	Cercle de confiance B
$rq3_3$	WS_Location	Cars_List	Write	Cercle de confiance C

TABLE 5.7 – Liste de requêtes $rq3_i$

5.6.4.3 Réception des permissions d'accès

Le tableau 5.8 représente les permissions d'accès des sous-requêtes ($rq3_1$), ($rq3_2$) et ($rq3_2$).

Sous-requête	**Service Web**	**permission d'accès**	**Cercle de confiance**
$rq3_1$	WS_Hotel	*deny*	cercle de confiance A
$rq3_2$	WS_Travel	untuned	Cercle de confiance B
$rq3_3$	WS_Leasing	refused	Cercle de confiance C

TABLE 5.8 – Permissions d'accès des requêtes rq_i

5.6.4.4 Adaptation des permissions d'accès

L'étape d'adaptation des permissions d'accès consiste à adapter les permissions issues des services Web appartenant aux cercles de confiances différents du cercle de confiance qui

utilise la politique du modèle de contrôle d'accès proposé. Les services Web (WS_Travel et WS_Leasing) appartiennent aux cercles de confiances B et C, ces cercles de confiances n'utilisent pas la politique de contrôle d'accès du modèle proposé, cela implique qu'avant d'utiliser ces permissions d'accès, il faut les adapter au modèle de contrôle d'accès proposé. La réalisation de cette étape implique l'utilisation de l'ontologie de domaine du médiateur. La figure 5.17 illustre les graphes des permissions d'accès ($rq3_2$ et $rq3_3$) après adaptation.

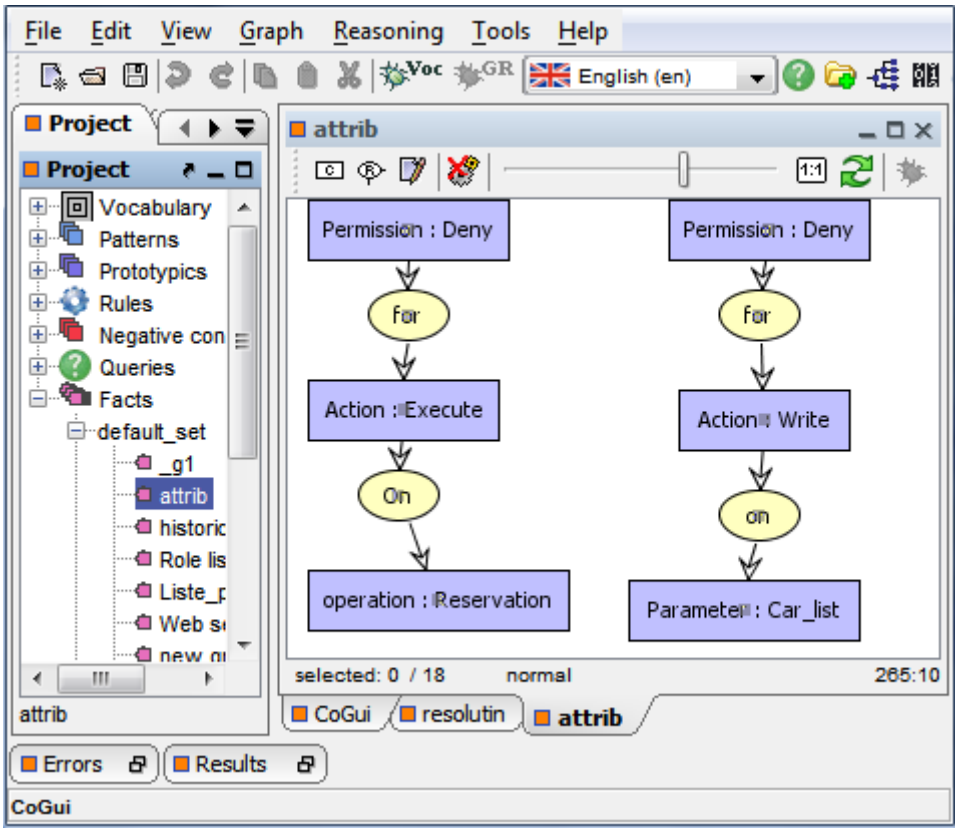

FIGURE 5.17 – Permissions d'accès après adaptations

5.6.4.5 Permission d'accès de la requête composite

Après l'adaptation des permissions d'accès, on constate que toutes les permissions d'accès sont négatives. Dans ce cas, la permission d'accès de la requête composite ($rq3$) est négative sans lancer le processus de négociation.

5.7 Conclusion

Dans ce chapitre, nous avons présenté les outils utilisés pour l'implémentation du prototype du modèle de contrôle d'accès proposé à savoir l'éditeur CoGui, la bibliothèque CoGITaNT et TooCom.

La réalisation de ce prototype est basée sur l'architecture des services Web et les communications entre les services Web se fait via des messages SOAP. Pour tester notre système, nous avons présenté trois cas, le premier cas représente un scénario d'une requête composite vers deux services Web issus d'un cercle de confiance qui utilise la politique du modèle de contrôle d'accès proposé, le deuxième cas représente un scénario similaire au premier, mais avec une négociation avec le sujet. Le troisième cas représente un scénario d'une requête composite vers trois services Web issus de cercles de confiances différents, avec une adaptation des permissions d'accès des services Web issus des cercles de confiances qui n'utilisent pas la politique du modèle de contrôle d'accès proposé. Et après avoir testé les différents scénarios d'exécution, le prototype répond favorablement aux exigences du modèle de contrôle d'accès proposé.

Conclusion générale

Dans ce travail, nous avons traité le problème du contrôle d'accès aux services Web composites. Le contrôle d'accès aux services Web est devenu actuellement un défi pour la communauté industrielle et académique. Plusieurs modèles proposés dans la littérature traitent le contrôle d'accès aux services Web. Néanmoins, peu d'entre eux supposent que la requête de l'utilisateur et une demande d'accès à un service Web composite.

Notre objectif consiste à proposer une spécification d'un modèle formel pour l'expression des autorisations d'accès aux Services Web.

Pour répondre à cet objectif, nous avons débuté notre travail par une étude bibliographique consacrée essentiellement au paradigme des services Web, l'étude des différents modèles de contrôle d'accès aux systèmes d'information et aux services Web, et l'étude des différentes approches formelles pour le contrôle d'accès. Cette étude nous a aidé à proposer un modèle de contrôle d'accès aux services Web composites décrit sémantiquement formalisé par les graphes conceptuels qui répond aux points suivants :

1. Nous avons utilisé un modèle de contrôle d'accès distribué où chaque service Web contrôle l'accès à ces ressources, selon sa propre politique de contrôle d'accès.
2. Au niveau de la politique de contrôle d'accès, nous nous sommes appuyés sur la notion de rôle, où l'attribution du rôle se fait par le profil utilisateur et chaque profil est construit à partir des attributs du sujet en utilisant une ontologie de domaine.
3. Du point de vue adaptation des permissions d'accès issues de différents services Web, nous avons utilisé une ontologie de domaine qui contient le vocabulaire des différentes permissions d'accès.
4. Du point de vue adaptation des attributs de l'utilisateur, nous avons utilisé une négociation entre le modèle de contrôle d'accès et le demandeur de la requête, afin que ce dernier puisse compléter les attributs manquants pour la satisfaction de sa requête.
5. Du point de vue représentation du modèle, nous avons utilisé les graphes conceptuels

pour la représentation et le raisonnement sur les différents éléments du modèle proposé.

6. Enfin, du point de vue architecture du système, nous avons implémenté un prototype du modèle de contrôle d'accès qui permet la vérification d'accès aux ressources demandées via une requête composite. Ce prototype se base sur l'architecture des services Web, où les messages échangés sont des messages SOAP, et le traitement de la requête se fait avec la participation des différents services Web impliqués dans la satisfaction de la requête.

Plusieurs autres extensions de ce travail sont possibles. En termes de perspectives possibles :

1. Traiter la résolution du conflit en utilisant les graphes conceptuels.
2. Élaborer un modèle formel de négociation.
3. Enfin, tester l'application à grande échelle.

Bibliographie

[1] A. Abou El Kalam, Y. Deswarte, and Amine Baïna et al. Access control for collaborative systems : A web services based approach. In *Proceedings of the IEEE International Conference on Web Services*, pages 1064–1071, Salt Lake City, Utah, USA, July 2007.

[2] S. Agarwal and B. Sprick. Specification of access control and certification policies for semantic web services. In *Proceedings of the E-Commerce and Web Technologies : 6th International Conference*, pages 348–357, Copenhagen, August 2005.

[3] R. Alur and D. L. Dill. The theory of timed automata. In *Proceedings of the Real-Time : Theory in Practice, REX Workshop*, volume 600, pages 45–73, June 1991.

[4] A. Anderson. Web services profile of xacml (ws-xacml). Technical report, OASIS, 2007.

[5] B. Bachimont. *Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances*. Eyrolles edition, 2000.

[6] A. Baina. *Contrôle d'Accès pour les Grandes Infrastructures Critiques : Application au réseau d'énergie électrique*. PhD thesis, Institut National des Sciences Appliquées, Toulouse, France, 2005.

[7] D.E. Bell and L.J. LaPadula. Secure computer systems : Mathematical foundations. *MITRE technical report, MITRE Corporation, Bedford Massachusetts*, 1973.

[8] E. D. Bell and J. L. La Padula. Secure computer system : Unified exposition and multics interpretation. Technical report, The MITRE Corporation, Bedford, M., 1976.

[9] F. Ben Jaâfar. *Une plate-forme de services Web gouvernementaux en ligne dans un environnement Web sémantique*. PhD thesis, Université de Laval, Québec, Canada, 2005.

[10] D. Berardi, D. Calvanese, and G. Giacomo et al. Automatic composition of e-services that export their behavior. In *Proceedings of the* 1^{st} *International Conference, Service-Oriented Computing - ICSOC*, pages 43–58, Trento, Italy, December 2003.

[11] D. Berardi, D. Calvanese, and G. Giacomo et al. Automatic composition of web services in colombo. In *Proceedings of the* 13^{th} *Italian Symposium on Advanced Database Systems*, pages 8–15, Brixen-Bressanone,Italy, June 2005.

[12] T. Berners-Lee and L. Kagal. The fractal nature of the semantic web. *AI Magazine*, 29(3) :29–34, 2008.

[13] E. Bertino, P. A. Bonatti, and et al. Trbac : A temporal role-based access control model. *ACM Transactions on Information and System Security*, 4(3) :191–233, 2001.

[14] E. Bertino, B. Catania, and E. Ferrari et al. A logical framework for reasoning about access control models. *ACM Transactions on Information and System Security*, 6(1) :71–127, February 2003.

[15] E. Bertino, L. Martino, and F. Paci et al. *Security for Web Services and Service-Oriented Architectures*. Springer edition, 2010.

[16] E. Bertino, A. C. Squicciarini, and L. Martino et al. An adaptive access control model for web services. *International Journal of Web Services Research*, 3(3) :27–60, 2006.

[17] R. Bhatti, E. Bertino, and A. Ghafoor. A trust-based context-aware access control model for web-services. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 184–191, June 2004.

[18] R. Bhatti, A. Ghafoor, and E. Bertino et al. X-gtrbac : an xml-based policy specification framework and architecture for enterprise-wide access control. *ACM Transactions on Information and System Security*, 8(2) :187–227, 2005.

[19] R. Bhatti, J. Joshi, and E. Bertino et al. Access control in dynamic xml-based web-services with x-rbac. In *Proceedings of the International Conference on Web Services*, pages 243–249, Las Vegas, Nevada, USA, June 2003.

[20] K. J Biba. Integrity considerations for secure computer systems. Technical report, MITRE Corp, 1977.

[21] S. Black and V. Varadharajan. A multilevel security model for a distributed object-oriented system. *IEEE Symposium on Security and Privacy*, 6(11) :68–78, 1990.

[22] D. Booth, H. Haas, and F. McCabe. Web services architecture. Technical report, W3C, 2003.

[23] D. F. C. Brewer and M. J. Nash. The chinese wall security policy. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 206–214, 1989.

[24] T. Bultan, X. Fu, and R. Hull et al. Conversation specification : a new approach to design and analysis of e-service composition. In *Proceedings of the* 12^{nd} *International World Wide Web Conference*, pages 403–410, 2003.

[25] E. Cerami. *Web services essentials*. O'Reilly edition, 2002.

[26] D. Chappell and T. JEWELL. *Java Web Service*. O'Reilly edition, 2002.

[27] S. Chatterjee and J. Webber. *Developing Enterprise Web Services*. Prentice Hall PTR edition, 2003.

[28] J. M. Chauvet. *Services Web avec SOAP, WSDL, UDDI, ebXML...* Eyrolles edition, 2002.

[29] P. Chen. The entity-relationship model toward a unified view of data. *ACM Transactions on Database Systems*, 1(1) :9–36, mar 1976.

[30] R. Chinnici and M. Gudgin. Web services description language. Technical report, 2004.

[31] D. B. Claro, P. Albers, and J. Hao. A framework for automatic composition of rfq web services. In *Proceedings of the IEEE International Conference on Services Computing - Workshops*, pages 221–228, Salt Lake City,Utah, USA, July 2007.

[32] P. Cloux, D. Doussot, and A. Géron. *Technologies et architectures Internet : Corbat, COM, XML, J2EE, .NET et Web Services.* Dunod edition, 2002.

[33] M. Couture. Description de java rmi et comparaison avec corba et dcom. Technical report, Université Laval, 2000.

[34] F. Cuppens and A. Miège. Adorbac : an administration model for or-bac. *Computer Systems Science and Engineering.*, 19(3), 2004.

[35] G. Dallons. Daml-s : interactions, critique et évaluation. Technical report, Institut d'informatique des FUNDP, Belgique, 2004.

[36] B. Daniela. *Automatique service composit models techniques and tools.* PhD thesis, Univarsité la sapienza, Rome, 2005.

[37] R. de la Rosa-Rosero. *Découverte et Sélection de Services Web pour une application Mélusine.* PhD thesis, Institut d'Informatique et de Mathématiques Appliquées de Grenoble, France, 2004.

[38] Marc Ehrig, Peter Haase, Mark Hefke, and Nenad Stojanovic. Similarity for ontologies - a comprehensive framework. In *Proceedings of the* 13^{th} *European Conference on Information Systems*, pages 1509–1518, Regensburg, Germany, May 2005.

[39] G. Gardien. *XML des bases de données aux Services Web.* Dunod edition, 2002.

[40] X. Gesnu. *Développer des Services Web XML et des composants serveurs.* Dunod edition, 2003.

[41] S. Godik and T. Moses. extensible access control markup language (xacml). Technical report, OASIS, 2003.

[42] K. D. Gottschalk, S. Graham, and H. Kreger. Introduction to web services architecture. *IBM Systems Journal*, 41(2) :170–177, 2002.

[43] T. R. Gruber. Automatically integrating heterogeneous ontologies from structured web pages. *Int. J. Semantic Web Inf. Syst.*, 3(1) :1–11, 2007.

[44] M. A. Harrison, W. L. Ruzzo, and et al. Protection in operating systems. *Commun. ACM*, 19(8) :461–471, 1976.

[45] P. Hoffmann. *Appariement contextuel d'ontologies*. PhD thesis, Université Claude Bernard, Lyon1, France, 2004.

[46] E. Hyvönen. *The semantic Web the new internet of meanings*. Institut de Technologie de l'Information Helsinki, Finland, 2002.

[47] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 31–42, Oakland, CA, USA, May 4-7 1997.

[48] K. Jensen. *Coloured Petri Nets : Basic Concepts,Analysis Methods and Practical Use*. Springer Verlag edition, 1997.

[49] J. Joshi, E. Bertino, and A. Ghafoor. An analysis of expressiveness and design issues for the generalized temporal role-based access control model. *IEEE Transactions on Dependable and Secure Computing*, 2(2) :157–175, 2005.

[50] H. Kadima. *Les Web Services*. Dunod edition, 2003.

[51] M. Koch, L. V. Mancini, and F. Parisi-Presicce. A graph-based formalism for rbac. *ACM Transactions on Information and System Security*, 5(3) :332–365, 2002.

[52] B. W. Lampson. Protection. *Operating Systems Review*, 8(1) :18–24, 1974.

[53] C. E. Landwehr. Formal models for computer security. *ACM Computing Surveys*, 13(3) :247–278, 1981.

[54] D. c. Lopes. *Etude et applications de l'approche MDA pour des plateformes de Services Web*. PhD thesis, UFR Sciences et Techniques Université de Nantes, France, 2005.

[55] E. Lozano, J. G. del Río, and J. Liem et al. Semantic feedback for the enrichment of conceptual models. In *Proceedings of the* 6^{th} *International Conference on Knowledge Capture*, pages 187–188, Banff, Alberta,Canada, June 2011.

[56] P. Lévy. Le futur web exprimera l'intelligence collective de l'humanité. *journal du net*, 2003.

[57] D. Martin, M. Burstein, and J. et al Hobbs. State of the art and state of the practice including initial possible research orientations. Technical report, InterOp,Interoperability Research for Networked Enterprises Applications and Software, 2006.

[58] D. Martin, M. Paolucci, and S. A. McIlraith et al. Bringing semantics to web services : The owl-s approach. In *Proceedings of the Semantic Web Services and Web Process Composition, First International Workshop*, pages 26–42, San Diego, USA, July 2004.

[59] T. Melliti. *Interopérabilité des Services Web complexes*. PhD thesis, Université Paris IX Dauphine, France, 2004.

[60] M. Mugnier. *Contributions algorithmiques pour les araphes d'héritage et les graphes conceptuels*. PhD thesis, Université Montpellier II, France, 1993.

[61] R. Neches, R. Fikes, and T. W. Finin et al. Enabling technology for knowledge sharing. *AI Magazine*, 12(3) :36–56, 1991.

[62] E. Newcomere. *Understanding Web Services Xml WSDL SOAP And UDDI*. O'Reilly edition, 2004.

[63] N. Nicolov, C. Mellish, and G. Ritchie. Sentence generation from conceptual graphs. In *Proceeding of the* 3^{rd} *International Conference on Conceptual Structures, ICCS*, August 1995.

[64] N. F. Noy and C. D. Hafner. The state of the art in ontology design : A survey and comparative review. *AI Magazine*, 18(3) :53–74, 1997.

[65] F. Paci, M. Mecella, and M. Ouzzani et al. Acconv - an access control model for conversational web services. *TWEB*, 5(3) :13, 2011.

[66] A-G. Perez, M. Fernandez, and O. Corcho. *Ontological Engineering*. Britich library edition, 2004.

[67] M. Pinheiro, M. Villanova-Oliver, and J. Gensel et al. Context-aware filtering for collaborative web systems : adapting the awareness information to the user's context. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, pages 1668–1673, Santa Fe, New Mexico, USA, March 2005.

[68] E. Salvat. *Raisonner avec des opérations de graphes : graphes conceptuels et règles d'inférence*. PhD thesis, Université Montpellier II, France, 2008.

[69] R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11) :9–19, 1993.

[70] R. S. Sandhu, E. J. Coyne, and et al. Role-based access control models. *IEEE Computer*, 29(2) :38–47, 1996.

[71] B. Shafiq, A. Masood, and J. Joshi et al. A role-based access control policy verification framework for real-time systems. In *Proceedings of international conference WORDS*, pages 13–20, Sedona,AZ, USA, February 2005.

[72] S. Short. *Construire des Services Web XML*. Dunod edition, 2002.

[73] Site. *Amine : une plate-forme pour le développement de systèmes et d'agents intelligents. Disponible sur l'URL : http ://sourceforge.net/projects/ amine-platform. Dernier consultation : Mars 2012.*

[74] Site. *CharGer : editeur de graphes conceptuels. Disponible sur l'URL : http ://sourceforge.net/projects/charger/. Dernier consultation : Mars 2012.*

[75] Site. *CoGITaNT : Conceptual Graphs Integrated Tools allowing Nested Typed graphs. Disponible sur L'URL : http ://cogitant.sourceforge.net. Dernier consultation : Mars 2012.*

[76] Site. *Cogui : outils de création graphique de graphes conceptuels. Disponible sur l'URL : http ://www.lirmm.fr/cogui. Dernier consultation : Mars 2012.*

[77] Site. *Consortium of standard Prosses for the Sharing of business Information (B2B). Disponible sur l'URL : http ://www.rosettanet.org/. Dernier consultation : Janvier 2012.*

[78] Site. *TooCom : a Tool to Operationalize an Ontology with the Conceptual Graph Model. Disponible sur l'URL : http ://sourceforge.net/projects/toocom/. Dernier consultation : Mars 2012.*

[79] Site. *Unified Modeling Language. disponible sur l'URL : http ://www.uml.org/, Dernier consultation : Mars 2012.*

[80] Site. *W3C : Ontolinga, A distributed collaborative Environnement to Browse, Create, Edit, Modify and Use Ontologies. Disponible sur L'URL : http ://www.ksl.stanford.edu/software/ontolingua/. Dernier consultation : Janvier 2012.*

[81] Site. *XML : Extensible Markup Language. Disponible sur l'URL : http ://www.w3.org/standards/xml/. Dernier consultation : Mars 2012.*

[82] J. F. Sowa. *Conceptual Structures : Information Processing in Mind and Machine.* Addison-Wesley edition, 1984.

[83] J F. Sowa. Conceptual graphs. *Knowl.-Based Syst.*, 5(3) :171–172, 1992.

[84] H. St-Louis. *Création d'une mémoire collective.* PhD thesis, Université Laval, Canada, 2008.

[85] R. K. Thomas. Team-based access control (tmac) : a primitive for applying role-based access controls in collaborative environments. In *Proceedings of the ACM Workshop on Role-Based Access Control*, pages 13–19, 1997.

[86] M. Uschold and M. Grüninger. Architectures for semantic integration. In *Proceedings of the Semantic Interoperability and Integration*, Dagstuhl, Germany, 2005.

[87] R. A. Whitehurst and T. F. Lunt. The sea view verification, northholland,amsterdam. In *Poceedings of the international conference CSFW*, pages 125–132, 1989.

[88] R. Wonohoesodo and Z. Tari. A role based access control for web services. In *Proceedings of the IEEE International Conference on Services Computing*, pages 49–56, Shanghai, China, September 2004.

[89] A. N. Zemirli. *Modèle d'accès personnalisé à l'information basé sur les Diagrammes d'Influence intégrant un profil utilisateur évolutif.* PhD thesis, Université Paul Sabatier de Toulouse III, France, 2008.

Annexe A : La bibliothèque CoGITaNT

Présentation de CoGITaNT

CoGITaNT (Conceptual Graphs Integrated Tools allowing Nested Typed graphs)) est une bibliothèque de classes C++ permettant le développement des logiciels basés sur le formalisme des graphes conceptuels. CoGITaNT est une extension de la plateforme CoGITo développée en 1994 par l'équipe de représentation des connaissances et d'inferences avec des graphes (LIRMM).

CoGITaNT fournit un grand nombre de classes et de méthodes élémentaires pour les graphes conceptuels. Ces dernières incluent la définition des types de concepts et de relations, la définition des règles de graphes, les graphes conceptuels emboités sont aussi supportés. La bibliothèque fournit aussi un certain nombre d'opérations pour manipuler les graphes conceptuels, la plus importante est l'opération de projection. Les bases de connaissances (supports ou graphes) peuvent êtres sauvegardés pour un usage ultérieur vu que la bibliothèque fournit des opérations d'entrées/sortie.

Une description détaillée de la bibliothèque est donnée dans les sections suivantes.

La hiérarchie des objets CoGITaNT

CoGITaNT suit une approche orientée objet, pour cela toutes ces fonctionnalités sont implémentées avec des classes.

La classe principale de CoGITaNT est Environment. Cette classe fournit des méthodes pour manipuler le support, exécuter des opérations de projection, appliquer les règles de graphes, écrire/lire la base de connaissances dans des fichiers sur disque et afficher tous les éléments de la base sur l'écran.

Une autre classe importante est la classe Graph. Cette classe gère les graphes qui sont représentés comme un ensemble d'instances de la classe Graph Object qui sont connectés avec un ensemble d'arcs (instances de la classe Edge). Évidemment, un n œud peut être un concept (la classe Concept) ou une relation (la classe Relation), mais pour représenter les graphes emboités, la classe Nesting peut être utilisée.

La classe Rule est utilisée pour représenter les règles. L'hypothèse et la conclusion d'une règle doivent être des graphes simples (et non pas emboîtés) et ne doivent pas avoir des liens de coréférence. La création des règles est tout à fait simple, puisque la classe fournit des méthodes pour la création des parties hypothèse et conclusion, aussi bien que pour définir des points de connexion possibles.

Une des classes de la bibliothèque CoGITaNT les plus fréquemment utilisées est l'iSet. Cette classe offre un identificateur pour chaque instance d'objet créé. Par exemple, le support contient un ensemble de types de concepts, l'environnement contient un ensemble de graphes et un ensemble de règles, chaque graphe contient un ensemble de concepts et un ensemble de relations, etc. La plupart des méthodes disponibles utilisent une instance de la classe iSet pour accéder à ces éléments.

Une partie importante de la hiérarchie de classes de CoGITaNT, chacune des classes est suivie par une courte description, est donnée ci dessus.

- **cogitant : :CogitantObject :** C'est la classe abstraite mère de n'importe qu'elle autre classe de la bibliothèque.
- **cogitant : : Environment :** L'environnement est l'objet qui contient le support, les graphes, et les règles définies sur ce support. En plus, cette classe fournit des méthodes pour utiliser facilement les opérations principales comme la projection, les opérations d'entrée/sortie (newGraph(), newRule(), graphs(), rules(), projections(), projectionsImage(), ruleApplications(), ruleApply(), disjoinSum(), readSupport(), writeSupport(), readGraph(), writeGraph()).
- **cogitant : : EnvironmentObject :** C'est la classe abstraite mère de chaque objet défini sur un environnement.
- **cogitant : : Graph :** Cette classe représente les graphes conceptuels en mémoire. Elle fournit des méthodes permettant la gestion des graphes (newGenericConcept(), newIndividualConcept(), newRelation(), link(), join()).
- **cogitant : : Rule :** Cette classe décrit une règle avec un graphe hypothèse, un graphe conclusion et lie entre les n œuds concepts de ces deux parties. L'hypothèse et la conclusion doivent être des graphes simples (hypothesis(), conclusion(), connectionPoints(), addConnectionPoint()).

- **cogitant : : GraphObject :** C'est la classe abstraite mère de chaque objet dans le graphe.
- **cogitant : : Concept :** Cette classe décrit les n œuds concepts dans un graphe.
- **cogitant : : Relation :** Cette classe décrit les n œuds relations dans un graphe.
- **cogitant : :ResultOpeProjection :** Cette classe représente le résultat de l'opération de projection (projections()) .
- **cogitant : : Support :** Cette classe représente le support sur lequel les graphes conceptuels sont construits (newConceptType(), newRelationType(), newIndividual()).
- **cogitant : :SupportObject :** C'est la classe abstraite mère de chaque objet défini dans le support.
- **cogitant : :Individual :** Cette classe est dédiée aux marqueurs individuels.
- **cogitant : :SupportType :** C'est la classe abstraite mère de chaque type dans le support.
- **cogitant : : ConceptType :** Cette classe permet la définition de types de concepts.
- **cogitant : :RelationType :** Cette classe permet la définition de types de relations.
- **cogitant : :Edge :** Cette classe représente un arc dans le graphe.
- **cogitant : : PartialOrder :** C'est une classe abstraite pour représenter un ordre partiel sur les instances de la classe iSet.

Annexe B : Description d'un service Web en OWL-S

La desc ription en OWL-S du service Web WS_hotel est représenté comme suit :
⟨ ?xml version = "1.0" encoding = "WINDOWS-1252" ? ⟩
⟨ rdf :RDF xmlns :owl = "http ://www.w3.org/2002/07/owl ♯ "
xmlns :rdfs = "http ://www.w3.org/2000/01/ rdf-schema ♯ "
xmlns :rdf = "http ://www.w3.org/1999/02/ 22-rdf-syntax-ns ♯ "
xmlns :service = "http ://www.daml.org/services/ owl-s/1.1/Service.owl ♯ "
xmlns :process = "http ://www.daml.org/services/ owl-s/1.1/Process.owl ♯ "
xmlns :profile = "http ://www.daml.org/services/ owl-s/1.1/Profile.owl ♯ "
xmlns :grounding = "http ://www.daml.org/services/ owl-s/1.1/Grounding.owl ♯ "

xml :base = "http ://127.0.0.1/services/1.1 /WS_hotel.owls" ⟩

⟨ owl :Ontology rdf :about = "" ⟩
⟨ owl :imports rdf :resource = "http ://127.0.0.1/ontology/Service.owl" / ⟩
⟨ owl :imports rdf :resource = "http ://127.0.0.1/ontology/Process.owl" / ⟩
⟨ owl :imports rdf :resource = "http ://127.0.0.1/ontology/Profile.owl" / ⟩
⟨ owl :imports rdf :resource = "http ://127.0.0.1/ontology/Grounding.owl" / ⟩
⟨ owl :imports rdf :resource = "http ://127.0.0.1/ontology/concept.owl" / ⟩
⟨ owl :imports rdf :resource = "http ://127.0.0.1/ontology/my_ontology.owl" / ⟩
⟨ owl :imports rdf :resource = "http ://127.0.0.1/ontology/travel.owl" / ⟩
⟨ /owl :Ontology ⟩

⟨ service :Service rdf :ID = "WS_hotel" ⟩
⟨ service :presents rdf :resource = " ♯ WS_HOTEL_PROFILE"/ ⟩
⟨ service :describedBy rdf :resource = " ♯ WS_HOTEL_PROCESS"/ ⟩
⟨ service :supports rdf :resource = " ♯ WS_HOTEL_GROUNDING"/ ⟩
⟨ /service :Service ⟩

⟨ profile :Profile rdf :ID = "WS_HOTEL_PROFILE" ⟩
⟨ service :isPresentedBy rdf :resource = " ♯ WS_hotel"/ ⟩
⟨ profile :serviceName xml :lang = "en" ⟩
SUBJECTREQUESTPERMISSION Hotel Service
⟨ /profile :serviceName ⟩
⟨ profile :textDescription xml :lang = "en" ⟩
This service returns permission for a given request to a given subject
and avaialbe in the particular hotel.
If hotel input is not given then it the condition to access.
⟨ /profile :textDescription ⟩
⟨ profile :hasInput rdf :resource = " ♯_REQUEST"/ ⟩
⟨ profile :hasInput rdf :resource = " ♯_SUBJECT"/ ⟩
⟨ profile :hasOutput rdf :resource = " ♯_PERMISSION"/ ⟩
⟨ profile :hasInput rdf :resource = " ♯_HOTEL"/ ⟩

⟨ profile :has_process rdf :resource = "WS_HOTEL_PROCESS" / ⟩
⟨ /profile :Profile ⟩

⟨!– ⟨ process :ProcessModel rdf :ID = "WS_HOTEL_PROCESS_MODEL" ⟩
⟨ service :describes rdf :resource = " ♯ WS_hotel"/ ⟩
⟨ process :hasProcess rdf :resource = " ♯ WS_HOTEL_PROCESS"/ ⟩
⟨ /process :ProcessModel ⟩
– ⟩

⟨ process :AtomicProcess rdf :ID = "WS_HOTEL_PROCESS" ⟩
⟨ service :describes rdf :resource = " ♯ WS_hotel"/ ⟩
⟨ process :hasInput rdf :resource = " ♯_REQUEST"/ ⟩
⟨ process :hasInput rdf :resource = " ♯_SUBJECT"/ ⟩
⟨ process :hasOutput rdf :resource = " ♯_PERMISSION"/ ⟩
⟨ process :hasInput rdf :resource = " ♯_HOTEL"/ ⟩
⟨ /process :AtomicProcess ⟩

⟨ process :Input rdf :ID = "_NUMBER" ⟩
⟨ process :parameterType rdf :datatype = "http ://www.w3.org/2001/XMLSchema
♯ anyURI" ⟩
http ://127.0.0.1/ontology/concept.owl ♯ NUMBER ⟨ /process :parameterType ⟩
⟨ rdfs :label ⟩
⟨ /rdfs :label ⟩

⟨ /process :Input ⟩

⟨ process :Output rdf :ID = "_RESERVATION" ⟩
⟨ process :parameterType rdf :datatype = "http ://www.w3.org/2001/XMLSchema
♯ anyURI" ⟩
http ://127.0.0.1/ontology/my_ontology.owl ♯ RESERVATION ⟨ /process :parameterType ⟩
⟨ rdfs :label ⟩
⟨ /rdfs :label ⟩
⟨ /process :Output ⟩

⟨ process :Output rdf :ID = "_CONFIRMATION" ⟩
⟨ process :parameterType rdf :datatype = "http ://www.w3.org/2001/XMLSchema
♯ anyURI" ⟩
http ://127.0.0.1/ontology/my_ontology.owl ♯ CONFIRMATION ⟨ /process :parameterType ⟩
⟨ rdfs :label ⟩
⟨ /rdfs :label ⟩
⟨ /process :Output ⟩

⟨ process :Input rdf :ID = "_HOTEL" ⟩
⟨ process :parameterType rdf :datatype = "http ://www.w3.org/2001/XMLSchema
♯ anyURI" ⟩
http ://127.0.0.1/ontology/travel.owl ♯ Hotel ⟨ /process :parameterType ⟩
⟨ rdfs :label ⟩
⟨ /rdfs :label ⟩
⟨ /process :Input ⟩

⟨ grounding :WsdlGrounding rdf :ID = "WS_HOTEL_GROUNDING" ⟩
⟨ service :supportedBy rdf :resource = " ♯ WS_hotel"/ ⟩
⟨ grounding :hasAtomicProcessGrounding ⟩
⟨ grounding :WsdlAtomicProcessGrounding rdf :ID =
"WS_HOTEL_AtomicProcessGrounding"/ ⟩
⟨ /grounding :hasAtomicProcessGrounding ⟩
⟨ /grounding :WsdlGrounding ⟩
⟨ grounding :WsdlAtomicProcessGrounding rdf :about =
" ♯ WS_HOTEL_AtomicProcessGrounding" ⟩
⟨ grounding :owlsProcess rdf :resource =
" ♯ WS_HOTEL_PROCESS"/ ⟩
⟨ grounding :wsdlInput ⟩
⟨ grounding :WsdlInputMessageMap ⟩

⟨ grounding :owlsParameter rdf :resource =
« ♯_HOTEL"/ ⟩
⟨ grounding :wsdlMessagePart rdf :datatype =
"http ://www.w3.org/2001/XMLSchema ♯ anyURI" ⟩
http ://127.0.0.1/wsdl/
HotelREQUESTSUBJECTPERMISSION ♯_HOTEL ⟨ /grounding :wsdlMessagePart ⟩
⟨ grounding :xsltTransformationString ⟩
None (XSL) ⟨ /grounding :xsltTransformationString ⟩
⟨ /grounding :WsdlInputMessageMap ⟩
⟨ /grounding :wsdlInput ⟩
⟨ grounding :wsdlDocument rdf :datatype =
"http ://www.w3.org/2001/XMLSchema ♯ anyURI" ⟩
http ://127.0.0.1/wsdl/WS_hotel.wsdl ⟨ /grounding :wsdlDocument ⟩
⟨ grounding :wsdlInput ⟩
⟨ grounding :WsdlInputMessageMap ⟩
⟨ grounding :owlsParameter rdf :resource = « ♯_NUMBER"/ ⟩
⟨ grounding :wsdlMessagePart rdf :datatype =
"http ://www.w3.org/2001/XMLSchema ♯ anyURI" ⟩
http ://127.0.0.1/wsdl/HotelHotelREQUESTSUBJECTPERMISSION
♯_NUMBER ⟨ /grounding :wsdlMessagePart ⟩
⟨ grounding :xsltTransformationString ⟩
None (XSL) ⟨ /grounding :xsltTransformationString ⟩
⟨ /grounding :WsdlInputMessageMap ⟩
⟨ /grounding :wsdlInput ⟩
⟨ grounding :wsdlInputMessage rdf :datatype =
"http ://www.w3.org/2001/XMLSchema
♯ anyURI" ⟩
http ://127.0.0.1/wsdl/HotelREQUESTSUBJECTPERMISSION ♯
get_RESERVATION_CONFIRMATIONRequest ⟨ /grounding :wsdlInputMessage ⟩
⟨ grounding :wsdlOutput ⟩
⟨ grounding :WsdlOutputMessageMap ⟩
⟨ grounding :owlsParameter rdf :resource = « ♯_CONFIRMATION"/ ⟩
⟨ grounding :wsdlMessagePart rdf :datatype =
"http ://www.w3.org/2001/XMLSchema ♯ anyURI" ⟩
http ://127.0.0.1/wsdl/
HotelREQUESTSUBJECTPERMISSION
♯_CONFIRMATION ⟨ /grounding :wsdlMessagePart ⟩
⟨ grounding :xsltTransformationString ⟩
None (XSL) ⟨ /grounding :xsltTransformationString ⟩

⟨ /grounding :WsdlOutputMessageMap ⟩
⟨ /grounding :wsdlOutput ⟩
⟨ grounding :wsdlOutput ⟩
⟨ grounding :WsdlOutputMessageMap ⟩
⟨ grounding :owlsParameter rdf :resource = « ♯_RESERVATION"/ ⟩
⟨ grounding :wsdlMessagePart rdf :datatype =
"http ://www.w3.org/2001/XMLSchema ♯ anyURI" ⟩
http ://127.0.0.1/wsdl/HotelNUMBERRESERVATIONCONFIRMATION
♯_RESERVATION ⟨ /grounding :wsdlMessagePart ⟩
⟨ grounding :xsltTransformationString ⟩
None (XSL) ⟨ /grounding :xsltTransformationString ⟩
⟨ /grounding :WsdlOutputMessageMap ⟩
⟨ /grounding :wsdlOutput ⟩
⟨ grounding :wsdlOperation ⟩
⟨ grounding :WsdlOperationRef ⟩
⟨ grounding :operation rdf :datatype =
"http ://www.w3.org/2001/XMLSchema ♯ anyURI" ⟩
http ://127.0.0.1/wsdl/HotelREQUESTSUBJECTPERMISSION
♯
get_RESERVATION_CONFIRMATION ⟨ /grounding :operation ⟩
⟨ grounding :portType rdf :datatype =
"http ://www.w3.org/2001/XMLSchema ♯ anyURI" ⟩
http ://127.0.0.1/wsdl/HotelREQUESTSUBJECTPERMISSION
♯ HotelREQUESTSUBJECTPERMISSIONSoap ⟨ /grounding :portType ⟩
⟨ /grounding :WsdlOperationRef ⟩
⟨ /grounding :wsdlOperation ⟩
⟨ grounding :wsdlOutputMessage rdf :datatype =
"http ://www.w3.org/2001/XMLSchema ♯ anyURI" ⟩
http ://127.0.0.1/wsdl/HotelREQUESTSUBJECTPERMISSION
♯ get_RESERVATION_CONFIRMATIONResponse ⟨ /grounding :wsdlOutputMessage ⟩
⟨ /grounding :WsdlAtomicProcessGrounding ⟩
⟨ /rdf :RDF ⟩

Résumé

Contrôler l'accès aux services Web hébergés dans de multiples sources de données distribuées est devenu actuellement un défi pour la communauté industrielle et académique. À ce jour aucun standard n'a été proposé pour les modèles de contrôle d'accès pour les services Web.

Dans ce mémoire, nous nous intéressons au contrôle d'accès aux services Web composites, et à sa spécification formelle par les graphes conceptuels. Le modèle proposé est distribué, il se base sur le concept de rôle. La particularité de ce modèle est que l'affectation des rôles se fait d'une manière automatique via le profil, et la sémantique utilisée dans le modèle permet l'adaptation des requêtes et de masquer l'hétérogénéité sémantique. Enfin, nous implémentons un prototype en utilisant la bibliothèque COGITANT afin de tester différents scénarios d'exécution.

Mots clés : Service Web, Sécurité, Contrôle d'accès, Graphe conceptuel, Ontologie.

Abstract

Control access to Web services hosted in multiple distributed data sources has now become a challenge for industry and academia communities. To date no standard has been proposed for models of access control for Web services.

In this dissertation we focus on access control for composite Web services, and its formal specification by conceptual graphs. The proposed model is distributed, it is based on the concept of role. The particularity of this model is that the role assignment is done automatically via a profile, and the semantics used in the model will fit requests and hide the semantic heterogeneity. Finally we implement a prototype using COGITANT librery to test different execution scenarios.

Keywords : Web Service, Security, Access Control, conceptual graph, Ontology.