République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mentouri – Constantine
Faculté des Sciences de l'Ingénieur – Département d'Informatique

N° d'ordre : 353/Mag/2009

N° de série : 010/Inf/2009

Mémoire

Pour l'obtention du diplôme de Magister en Informatique

Thème :

Construction d'une ontologie pour le domaine de la sécurité : Application aux agents mobiles

Présenté par :
Riad LEKHCHINE

Dirigé par :
Pr. Zizette BOUFAÏDA

Composition du Jury :

Pr. Mahmoud BOUFAÏDA	Université de Constantine	Président
Pr. Zizette BOUFAÏDA	Université de Constantine	Rapporteur
Dr. Salah MERNIZ	M.C.A Université de Constantine	Examinateur
Dr. Nadia ZEGHIB	M.C.A Université de Constantine	Examinateur
Dr. Salima HACINI	M.C.B Université de Constantine	Invitée

Année Universitaire 2008-2009

Dédicace

A mes parents,

ma source d'existence

Remerciements

Tout d’abord, je commence par remercier de tout mon cœur ma chère honorable encadreur Zizette BOUFAÏDA qui m’a soutenu et m’a été plus qu’un encadreur.

Je tiens aussi à remercier vivement mon co-encadeur Salima HACINI.

Je tiens à présenter de tout mon cœur mes remerciements et ma reconnaissance au Professeur Mahmoud BOUFAÏDA pour avoir accepté de présider mon jury de soutenance.

Ensuite, Je souhaite remercier Salah MERNIZ et Nadia ZEGHIB pour m’avoir accepté d’évaluer et juger mon travaille et d’avoir pris part de mon jury.

Je passe une salutation respectueuse pour tout enseignant qui a contribué à l’obtention du titre Magiser en informatique.

Ainsi que tous qui ont été, de prés ou loin, de précieuse aide.

A tous ceux que j’aime et que je respecte.

Les derniers remerciements vont à ma famille pour tout ce qu’elle a fait pour moi et sans laquelle, rien de ce qui est entre vos mains aujourd’hui n’aurait été réalisé.

ملخــص:

أصبحت الأنظمة المفتوحة بحاجة إلى المزيد من المرونة والفعالية من أجل إنجاح الاتصال والتفاعل بين مختلف الوحدات، خصوصا في ما يتعلق بمسألة الأمن، فهذا الأخير يعتبر عاملا رئيسيا في التجارة الإلكترونية وبالأخص في بيئة تبادل تجاري مكونة من خدمات مفتوحة غير متجانسة، فتنوع مقاييس الأمن المستعملة من شأنه عرقلة التعاون الأمني.

إن مشكلة الأمن المتعلقة بالعميل المتحرك تحد من استخدام تكنولوجيا العملاء المتحركين وتعيق توسعها وتطبيقها فالتزايد المستمر للتعقيد والتوسع على مستوى الأنظمة والتطبيقات يزيد من صعوبة وضع نظام أمن موحد وسياسة أمنية مسبقة.

إن الأنتولوجيا تعتبر من بين أهم الحلول لمشاكل اللاتجانس، فهي توفر سيميائية مشتركة من شأنها أن تمنع فشل الاتصال والتفاعل بين العملاء المتحركين الناتج عن عدم تجانس في خصائصهم الأمنية.

هدفنا الرئيسي في هذه المذكرة هو بناء أنتولوجيا لمجال الأمن وتطبيقها على العملاء المتحركين باتباع مخطط بناء والاعتماد على معلومات تتعلق بالخصائص والإجراءات الأمنية التي يستخدمها العميل المتحرك والمضيف.

كلمات مفتاحية : أنتولوجيا الأمن ، أمن العميل المتحرك.

Résumé

Actuellement, les systèmes ouverts exigent davantage de flexibilité et d'efficacité afin de réussir la communication et l'interaction entre différentes entités, en particulier en ce qui concerne la question sécuritaire. Ce dernier est considéré comme le facteur principal dans le commerce électronique, notamment dans un environnement des échanges commerciaux constitué de services ouverts hétérogènes, car la diversité des standards de sécurité utilisés peut entraver l'interopérabilité sécuritaire.

Le problème de sécurité concernant l'agent mobile limite l'utilisation de la technologie des agents mobiles et entrave son extension et application, car la progression continue de la complexité et de l'extensibilité au niveau des systèmes et applications dans l'environnement dynamique augmente la difficulté d'appliquer un système sécuritaire unifié ainsi qu'une politique de sécurité anticipée.

L'ontologie est considérée parmi les solutions les plus importantes de problème d'hétérogénéité, car elle offre une sémantique partagée capable d'empêcher l'échec de communication et d'interaction entre les agents mobiles dû à l'hétérogénéité de leurs propriétés sécuritaires.

Notre but principal dans ce mémoire est la construction d'une ontologie pour le domaine de la sécurité et son application aux agents mobiles selon un processus de construction et en se basant sur des informations concernant les propriétés et procédés sécuritaires utilisées par l'agent mobile et l'hôte.

Mots clés : Ontologie de sécurité, sécurité d'agent mobile.

Abstract

Nowadays, open systems require more flexibility and efficiency in order to succeed the communication and interaction among various components, particularly, concerning security problem. This latter is considered as a major driver for the success of E-business, especially in B2B environment with heterogeneous open services.

The mobile agent security problem limits the use of mobile agent technology and hinders its extensibility and its application. Since the increasing of complexity and extension at system and applications level increase the difficulty to implement a common security system and security policy in advance.

Ontology is considered among the most important solution to heterogeneity, as it offers a shared knowledge which is able to prevent communication and interaction failure among mobile agents, this failure is due to their heterogeneous security properties.

Our main aim, in this research work, is to build ontology for the security domain, and its application to mobile agents following a building process based on the information concerning the security properties and procedures used by mobile agent and host.

***Keywords**: Security ontology, Mobile agent security.*

Table des matières

Introduction générale

1 CONTEXTE GÉNÉRAL 2

2 PROBLÉMATIQUE 2

3 MOTIVATION 3

4 NOTRE CONTRIBUTION 3

5 PLAN DE MÉMOIRE 4

Chapitre 1 - La sécurité des agents mobiles

1 INTRODUCTION 6

2 LES AGENTS MOBILES 6

2.1 DEFINITION D'AGENT : 6

2.2 MODES D'EXECUTION : 7

2.2.1 Client/Serveur : *7*

2.2.2 Exécution à distance : *7*

2.2.3 Code à la demande : *7*

2.2.4 Agents mobiles : *8*

2.3 AGENT MOBILE : DEFINITION, CARACTERISTIQUES ET PROPRIETES : 8

2.3.1 Définition : *8*

2.3.2 Caractéristiques : *8*

2.3.3 Propriétés d'agent mobile : *9*

2.4 DOMAINES D'APPLICATION : 11

2.4.1 Commerce électronique : *11*

2.4.2 Applications de surveillance : *11*

2.4.3 Actions ambulantes : *11*

2.4.4 Configuration de système : *11*

2.4.5 Traitement en parallèle : *11*

2.4.6 Rassemblement de l'information : *11*

3 SÉCURITÉ INFORMATIQUE 12

3.1 DEFINITION DE LA SECURITE INFORMATIQUE : 12

3.2 POLITIQUE DE SECURITE : 12

3.3 LES RISQUES : 12

3.4 LES ATTAQUES : 13

3.5 LES PROCEDES TECHNIQUES DE PROTECTION : 13

3.5.1 Cryptographie (la science du secret) : *13*

3.5.2 Signature numérique : *14*

3.5.3 Certificat numérique : *15*

3.5.4 Infrastructure de Gestion de Clés ICG : *15*

4 SÉCURITÉ POUR LES AGENTS MOBILES 16

4.1 EXIGENCES DE SECURITE POUR LES AGENTS MOBILES : 16

4.1.1 Confidentialité *16*

4.1.2 Intégrité *16*

4.1.3 Responsabilité *17*

4.1.4 Disponibilité ... *17*
4.1.5 Anonymat ... *18*
4.2 PROBLEMES DE SECURITE D'AGENT MOBILE : ... 18
4.2.1 Exécution d'agent : ... *18*
4.2.2 Autonomie : ... *18*
4.2.3 Communication : ... *18*
4.2.4 Rationalité, véracité, et bienveillance : ... *19*
4.3 MENACES DE SECURITE : ... 19
4.3.1 Menaces d'agent contre hôte : ... *19*
4.3.2 Menaces d'agent contre agent : ... *20*
4.3.3 Menace de l'hôte contre agent : ... *21*
4.4 CONTRE-MESURES : ... 22
4.4.1 Protection des agents mobile : ... *22*
4.4.2 Protection des hôtes : ... *25*
5 CONCLUSION ... 27

Chapitre2 - L'Ingénierie Ontologique

1 INTRODUCTION ... 29
2 NOTION D'ONTOLOGIE ... 29
2.1 DEFINITIONS ... 29
2.2 QUE REPRESENTE-T-ON DANS UNE ONTOLOGIE ? ... 30
2.3 DIFFERENTES SORTES D'ONTOLOGIES ... 32
2.3.1 Objet de conceptualisation ... *32*
2.3.2 Niveau de formalisme de représentation ... *32*
2.4 LES ONTOLOGIES ET LES SYSTEMES A BASES DE CONNAISSANCES ... 33
2.5 LES ONTOLOGIES ET LE WEB SEMANTIQUE ... 33
3 LES ONTOLOGIES : DIFFERENTS BESOINS ... 34
3.1 COMMUNICATION : ... 34
3.2 INTEROPERABILITE : ... 34
3.3 INGENIERIE DES SYSTEMES : ... 35
4 UN SQUELETTE DE METHODOLOGIE POUR CONSTRUIRE DES ONTOLOGIES ... 35
4.1 EVALUATION DES BESOINS ... 35
4.2 CONCEPTUALISATION : ... 36
4.3 ONTOLOGISATION : ... 36
4.4 OPERATIONNALISATION ... 36
5 QUELQUES METHODOLOGIES DE CONSTRUCTION D'ONTOLOGIES ... 36
5.1 TOVE : ... 37
5.2 ENTERPRISE : ... 37
5.3 METHONTOLOGY : ... 38
5.3.1 Spécification : ... *38*
5.3.2 Conceptualisation : ... *38*
5.3.3 Implémentation : ... *38*
5.3.4 Maintenance : ... *38*
6 FORMALISMES DE REPRÉSENTATION ... 38
6.1 FRAMES : ... 39

6.2 GRAPHES CONCEPTUELS : 39
6.3 LOGIQUE DE DESCRIPTION LD : 40
6.3.1 Historique : 40
6.3.2 Les deux niveaux de description : 40
6.3.3 La logique A L : 41
6.3.4 Les extensions d' A L : 41
6.3.5 L'inférence : 43
7 OUTILS DE DEVELOPPEMENT D'ONTOLOGIES 43
7.1 LANGAGES DE SPECIFICATION D'ONTOLOGIES : 43
7.1.1 RDF : 43
7.1.2 RDF(S) : 44
7.1.3 DAML-OIL : 44
7.1.4 OWL : 45
7.2 LES MOTEURS D'INFERENCES : 46
7.2.1 Racer : 46
7.2.2 Pellet : 47
7.3 LANGAGES D'INTERROGATION D'ONTOLOGIES : 47
7.3.1 RDQL : 47
7.3.2 SPARQL : 48
7.3.3 nRQL : 48
7.4 EDITEUR D'ONTOLOGIES : 48
7.4.1 OILEd 49
7.4.2 OntoEdit 49
7.4.3 Ontolingua 49
7.4.4 ONTOSAURUS 49
7.4.5 Protégé2000 49
8 LES ONTOLOGIES DANS LA SECURITE INFORMATIQUE 50
8.1 CONTROLE D'ACCES A BASE DE ROLES (RBAC) : 50
8.1.1 Classes d'entités : 50
8.1.2 Relations RBAC0 : 51
8.1.3 Fonctions RBAC0 : 51
8.1.4 RBAC1 et R B A C 2 : 51
8.2 KAOS : 51
8.2.1 Politique de KAoS : 51
8.2.2 Ontologie de politique : 52
8.3 REI : 53
8.4 DISPOSITIFS DES DIFFERENTES APPROCHES : 54
9 CONCLUSION 54

Chapitre 3 - Construction de l'ontologie

1 INTRODUCTION 56
2 L'ONTOLOGIE MASO 57
2.1 CONTEXTE DE SECURITE : 57
2.2 UTILISATION DE L'ONTOLOGIE : 57
2.2.1 Communication entre deux agents mobiles : 58
2.2.2 Communication entre plusieurs entités : 59
2.3 SCENARIOS DE COMMUNICATION : 59

2.3.1 Scénario 1 : ... *59*
2.3.2 Scénario 2 : ... *60*

3 PROCESSUS DE CONSTRUCTION DE L'ONTOLOGIE ... 61

3.1 SPECIFICATION : ... 61
3.2 CONCEPTUALISATION : ... 61
3.3 FORMALISATION : ... 62
3.4 IMPLEMENTATION : ... 62
3.5 TESTS ET EVOLUTION ... 62

4 CONSTRUCTION D'UNE ONTOLOGIE POUR LE DOMAINE DE LA SÉCURITÉ ... 63

4.1 SPECIFICATION : ... 63
4.2 CONCEPTUALISATION : ... 64
4.2.1 Construction de glossaire de termes : ... *65*
4.2.2 Construction du diagramme de classification de concepts : ... *69*
4.2.3 Construction de diagramme de relations binaires : ... *71*
4.2.4 Dictionnaire de concepts : ... *72*
4.2.5 Tableaux des relations binaires : ... *75*
4.2.6 Tableaux des attributs : ... *76*
4.2.7 Tableaux des axiomes logiques : ... *77*
4.2.8 Tableaux des instances : ... *78*
4.3 FORMALISATION : ... 79
4.3.1 Construction de TBox : ... *79*
4.3.2 Construction de ABox : ... *82*
4.4 IMPLEMENTATION : ... 84
4.4.1 Présentation de l'éditeur PROTEGE OWL : ... *84*
4.4.2 Définition de la hiérarchie des classes : ... *85*
4.4.3 Définition des propriétés : ... *86*
4.4.4 Définitions des restrictions : ... *89*
4.4.5 Définition des instances : ... *90*
4.4.6 Génération du code : ... *90*
4.5 TEST DE L'ONTOLOGIE : ... 90

5 CONCLUSION ... 92

Conclusion générale & perspectives

1 CONCLUSION ... 95

2 CONTRIBUTION ... 95

2.1 DEVELOPPEMENT DES SCENARIOS DE COMMUNICATION ENTRE LES AGENTS MOBILES ET LES PLATES-FORMES : ... 95
2.2 CONSTRUCTION D'UNE ONTOLOGIE DE DOMAINE DANS LE CADRE DE LA SECURITE DES AGENTS MOBILES : ... 95

3 EVOLUTION & PERSPECTIVES ... 96

3.1 EVOLUTION DES SCENARIOS DE COMMUNICATION PROPOSES : ... 96
3.2 EVOLUTION DE L'ONTOLOGIE DE DOMAINE : ... 96

REFERENCES BIBLIOGRAPHIQUES : ... 97

SITES WEB : ... 103

GLOSSAIRE : ... 104

ANNEXE ... 105

Table des Figures

Figure 1 – Schéma d'organisation Client/Serveur....7
Figure 2 – Envoi du savoir faire....7
Figure 3 – Récupération du savoir faire....7
Figure 4 – Les couches du Web sémantique....33
Figure 5 – Domaines d'utilisation des Ontologies....34
Figure 6 – Processus de construction d'ontologie....35
Figure 7 – La pyramide des langages du Web sémantique....43
Figure 8 – Taxinomie de concept personne....44
Figure 9 – Diagramme représentant l'ontologie KAoS....52
Figure 10 – Une partie de l'ontologie Rei....53
Figure 11 – Agent mobile sensible au contexte de sécurité....57
Figure 12 – Communication entre deux agents mobiles en utilisant l'ontologie MASO....58
Figure 13 – Communication avec l'ontologie MASO dans un environnement dynamique....59
Figure 14 – Scénario de communication 1 entre agent mobile et plate-forme....59
Figure 15 – Scénario de communication 2 entre agent mobile et plate-forme....60
Figure 16 - Un document RDF de spécification de l'ontologie.....64
Figure 17 – Diagramme de classification de concepts....70
Figure 18 – Diagramme de relations binaires....71
Figure 19 – Interface de Protégé OWL....84
Figure 20 – Création des classes....85
Figure 21 – Création des propriétés pour une classe....86
Figure 22 – Création d'un attribut....87
Figure 23 – Création d'une relation....88
Figure 24 – Création d'une restriction sur une classe....89
Figure 25 – Création des instances....90
Figure 26 – Vérification de l'URL de Reasoner....91
Figure 27 – Lancement de Racer....91
Figure 28 – Test de consistance....92
Figure 29 – Test de classification....92

Liste des Tableaux

Tableau 1 – Comparaison entre les différents modes d'exécution 8
Tableau 2 – Une base de connaissances composée d'une TBox et d'une ABox 40
Tableau 3 – Les constructeurs selon AL 41
Tableau 4 – Les constructeurs des LDs 42
Tableau 5 – Les constructeurs de OWL Lite 45
Tableau 6 – Liste des constructeurs ajoutés par OWL DL 46
Tableau 7 – Les principaux moteurs d'inférence 46
Tableau 8 – Liste de domaines, ontologies et classes de Rei 53
Tableau 9 – Glossaire de termes 68
Tableau 10 – Dictionnaire de concepts 74
Tableau 11 – Tableau des relations binaires 75
Tableau 12 – Tableau des attributs 76
Tableau 13 – Tableau des axiomes logiques 77
Tableau 14 – Tableau des instances 78
Tableau 15 – définition de la TBox 82
Tableau 16 – Partie assertionnelle des concepts 83
Tableau 17 – Partie assertionnelle des relations 83

INTRODUCTION GÉNÉRALE

1 CONTEXTE GÉNÉRAL

Le commerce électronique (*e-business*) fait actuellement face aux modifications révolutionnaires : les marchés électroniques permettent de nouveaux genres de services et d'interactions entre les fournisseurs et les clients. C'est un domaine d'application où des milliers d'entreprises hétérogènes, agissent en tant que fournisseurs et/ou clients.

Depuis le début des années 90, l'utilisation des agents mobiles dans les systèmes distribués et spécialement dans le cadre de commerce électronique B2B *(Business To Business)* est étudiée. Aujourd'hui, la technologie « agent mobile » est de plus en plus attractive vu les avantages qu'elle offre pour la réalisation des tâches complexes dans les systèmes ouverts et dynamiques.

Cependant, un frein à l'utilisation réelle et étendue de cette technologie réside dans les problèmes de sécurité qu'elle introduit. La mise en place d'une politique de sécurité peut demander, d'une part la protection des ressources et des données des machines hôtes, et d'autre part, la préservation de l'intégrité et la confidentialité des agents eux-mêmes et de leurs communications.

Dans ce contexte, l'intérêt autour de la protection et la sécurité des agents mobiles a augmenté au sein des organismes. Par conséquent, différentes politiques de sécurité ont été développées, et différentes normes de sécurité ont été proposées. Ceci a engendré une hétérogénéité dans l'exploitation de ces politiques de sécurité par des entités différentes.

En plus, La croissance de l'utilisation des systèmes multi-agents SMA dans les systèmes distribués a déclenché des opportunités pour le calcul coopératif et l'interopérabilité électronique « e-interopérabilité », où les entreprises conduisent conjointement le calcul et partagent des tâches d'affaires entre leurs ressources. Ces e-interopérabilités se produisent généralement entre des organismes ayant une politique de sécurité différente.

Durant la dernière décennie, l'usage des ontologies s'est accru pour résoudre le problème de l'interopérabilité sémantique. Un des plus grands projets basés sur l'utilisation d'ontologies est le Web sémantique qui consiste à ajouter au Web actuel une véritable couche de connaissances permettant, dans un premier temps, des recherches d'information au niveau sémantique et non plus syntaxique. A terme, il est prévu que des applications Internet pourront mener des raisonnements utilisant les connaissances stockées sur la Toile.

2 PROBLÉMATIQUE

L'aspect sémantique porte sur l'étude du sens des entités et de leur relation indépendamment de la façon de les représenter. L'hétérogénéité des modèles de représentation, des langages de programmation, des systèmes d'exploitation ainsi que des mécanismes d'échange et de protection des données, implique généralement une mauvaise interprétation.

Si nous fonctionnons dans un environnement homogène fermé, nous pouvons spécifier et convenir l'utilisation d'une configuration de sécurité statique. Cependant, dans un environnement ouvert et hétérogène, la variété d'utilisations de normes et politiques de sécurité peut gêner l'interopérabilité de sécurité parce qu'une configuration de sécurité commune ne peut pas toujours être convenue à l'avance, découverte dynamiquement ou

modifiée dynamiquement. De ce fait, si nous fonctionnons dans un domaine hétérogène, nous avons besoin d'une approche plus dynamique pour configurer et atteindre une sécurité.

Afin de réaliser l'interopérabilité et résoudre les questions d'hétérogénéité de la politique de sécurité des agents mobiles, l'intégration sémantique est nécessaire. Les ontologies, à l'heure actuelle, sont considérées en tant que la prochaine tendance pour résoudre les problèmes d'hétérogénéité. C'est la raison qui nous à poussé à produire une ontologie de domaine de sécurité commune qui présentera des concepts, relations, contraintes d'intégrité et des règles sur lesquelles les agents et hôtes pourraient collaborer.

Les agents d'entreprise (agents clients et agents fournisseurs) sont des entités autonomes, potentiellement hétérogènes, d'origines diverses, et libres d'entrer et sortir du système quand ils le souhaitent. Dans un tel scénario, on rencontre fréquemment des problèmes d'interopérabilité demandant des techniques spécifiques de résolution. Notre effort se concentre sur la résolution de ce genre de problèmes en se limitant sur l'hétérogénéité des politiques de sécurité.

L'objectif de ce mémoire consiste en la construction d'une ontologie de domaine de sécurité afin d'éliminer les différences sémantiques qui existent au niveau des objets, attributs, et structures de données de politiques de sécurité pour faciliter l'interopérabilité des agents mobiles.

3 MOTIVATION

Le manque d'une solution commune qui harmonise les divers modèles de sécurité est un obstacle majeur dans le développement des systèmes ouverts, et en particulier le commerce électronique inter-entreprises.

La sécurité est une large discipline, et il n'y a aucun ensemble standard de caractéristiques de sécurité qui sera optimum à travers des domaines hétérogènes d'application.

Il n'y a aucune terminologie commune pour la sécurité à travers les différents domaines d'application. En plus, il n'existe aucune architecture ouverte commune pour la sécurité des agents mobiles.

4 NOTRE CONTRIBUTION

Notre contribution dans ce contexte consiste en la construction d'une ontologie de domaine de sécurité pour les applications et les systèmes utilisant la technologie d'agent mobile. Le développement de cette ontologie doit suivre un processus de construction d'ontologies comptant un ensemble de phases spécifiées de façon très détaillée afin de garantir un bon vocabulaire et d'aboutir à une ontologie typique. Pour cela, nous nous attachons dans ce mémoire à construire l'ontologie en suivant les étapes d'un processus de développement d'ontologie partant de connaissances brutes et arrivant à une ontologie de domaine représentée par le langage OWL.

Nous suggérerons, en outre, et de manière superficielle, des scénarios de communication entre les agents mobiles afin d'exploiter l'ontologie construite et d'avoir une interopérabilité de sécurité plus compréhensible.

5 PLAN DE MÉMOIRE

Pour présenter notre travail, nous suggérons le plan de lecture suivant :

Le premier chapitre est dédié à la présentation du domaine de sécurité des agents mobiles. Il commence par la définition de la notion d'agent mobile, les propriétés et le domaine d'application. Ensuite, il présente la notion de sécurité, les exigences et problèmes de sécurité dans le domaine d'agent mobile, et les principales approches utilisées pour protéger l'agent mobile.

Le deuxième chapitre est consacré à la présentation des ontologies. Nous commençons par la définition de la notion d'« ontologie » en Ingénierie des Connaissances. Nous présentons ensuite les principaux formalismes de représentation de connaissances, à savoir, les frames, les graphes conceptuels et les logiques de descriptions. Nous découvrirons après les méthodologies les plus représentatives de leur construction et quelques domaines de leur utilisation. A la fin, nous présenterons les outils nécessaires à leur développement, à savoir, les langages de représentation, les outils d'édition et d'interrogation, …etc.

Le troisième chapitre concerne la présentation en détail de notre travail, dans lequel nous avons utilisé le processus choisi pour arriver à la construction de l'ontologie d'application.

Le mémoire s'achève avec une conclusion générale récapitulant le contexte de recherche de notre étude, la démarche suivie, nos contributions et énonce un ensemble de perspectives.

CHAPITRE 1

La SÉCURITÉ DES AGENTS MOBILES

1 INTRODUCTION

Le paradigme agent mobile est proposé comme une solution prometteuse facilitant l'exécution distribuée à travers un réseau ouvert. Il offre une panoplie d'avantages **[Lange99]** tels qu'une exécution asynchrone et autonome, une diminution du trafic sur le réseau, une tolérance aux fautes ou encore une réduction du temps d'exécution des tâches. L'agent mobile améliore les performances puisqu'il permet de transférer les unités d'exécution vers les clients plutôt que de capturer les données pour les traiter localement. Cependant, le déplacement d'un agent mobile à travers le réseau n'est pas toujours sans risques. En effet, l'agent peut être attaqué par un autre agent, par une personne connectée au réseau ou par un site (hôte ou plate-forme) visité sur lequel l'agent devra s'exécuter **[Hohl97]**. Cette technologie a introduit plusieurs problèmes sérieux tels que le problème de l'hétérogénéité, celui du maintien des communications ou encore celui de la gestion des ressources partagées **[Vigna04**, **Chess94]**. De plus, elle a accentué les effets des problèmes de sécurité existants **[Bella04**, **Borselius02**, **Farmer96**, **Karnik98**, **Reiser00**, **Roth99**, **Vigna98]**.

Ce chapitre introduit le concept d'agent mobile en mettant l'accent sur le problème de la sécurité que son utilisation engendre. Il décrit les techniques les plus importantes développées pour sa protection.

2 LES AGENTS MOBILES

2.1 Définition d'agent :

« Un agent est un système informatique *situé* dans un environnement, et qui agit d'une façon *autonome* et *flexible* pour atteindre les objectifs pour lesquels il a été conçu. » **[Jennings98]**.

On appelle agent une entité physique ou virtuelle : **[ht1]**

- qui est capable d'agir dans un environnement.
- qui peut communiquer directement avec d'autres agents.
- qui est mue par un ensemble de tendances (sous forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser).
- qui possède des ressources propres.
- qui est capable de percevoir (mais de manière limitée) son environnement.
- qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune).
- qui possède des compétences et offre des services.
- qui peut éventuellement se reproduire.
- dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

Un système multi-agent (*Multi Agent System MAS*) est un système composé d'agents autonomes multiples avec les caractéristiques suivantes **[Jennings98]** :

- Un agent ne peut pas résoudre un problème sans aide.
- Il n'y a aucun controle du système global.
- Les données sont décentralisées.
- Le calcul est asynchrone.

2.2 Modes d'exécution :

Pour la conception d'application distribuée, trois modèles principaux étendent le paradigme client/serveur pour exploiter la mobilité du code : l'exécution à distance (*Remote Evaluation REV*), le code à la demande (*Code on Demand CoD*) et les agents mobiles **[Carzaniga97]**.

2.2.1 Client/Serveur :

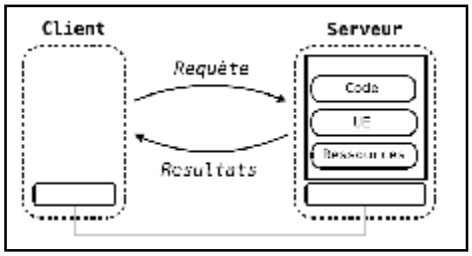

Figure 1 – Schéma d'organisation Client/Serveur

C'est le paradigme le plus répandu aujourd'hui pour l'informatique distribuée. Dans ce paradigme, un serveur est défini comme unité d'exécution (UE) qui se trouve sur un site SB et offre un ensemble de services, des ressources et un savoir-faire (Code). Le client A se trouvant sur un site SA demande l'exécution des services par l'interaction avec le serveur SB. Après l'exécution des services demandés, le résultat est délivré au client. Le serveur fournit donc à la fois les ressources et le savoir-faire.

2.2.2 Exécution à distance :

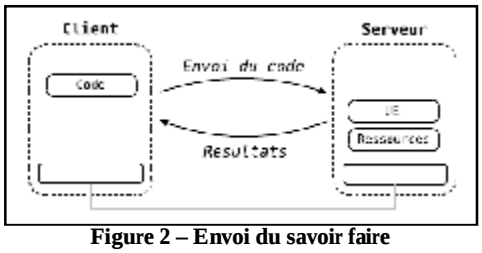

Figure 2 – Envoi du savoir faire

Dans ce mode, le client A détient le savoir-faire (Code) et le serveur B détient les ressources. Par conséquent, A envoie le code à B pour qu'il puisse l'exécuter en utilisant ses ressources, et renvoie le résultat à A.

2.2.3 Code à la demande :

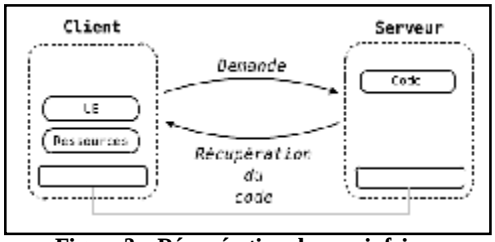

Figure 3 – Récupération du savoir faire

Dans ce mode, le client A possède les ressources, mais n'a pas le savoir-faire nécessaire pour le traitement. Il doit alors interagir avec le serveur B. Lorsque A reçoit le code, il peut l'utiliser avec ses ressources.

2.2.4 Agents mobiles :

Ce paradigme est une extension de paradigme Exécution à distance. Le dernier consiste à transférer le code, mais l'agent mobile implique la mobilité de l'entité de calcul entière, avec son code, état, et parfois avec ses propres ressources. Un agent mobile A migre vers SB avec son code et poursuit l'exécution en utilisant les ressources de SB.

Le tableau ci-dessous compare les différents modes.

Paradigme	Avant		Après	
	SA	SB	SA	SB
Client/serveur	A	code, res, B	A	code, res, B
Exécution à distance	code, A	res, B	A	code, res, B
Code à la demande	res, A	code, B	code, res, B	B
Agent mobile	code, A	res	-	code, res, A

Tableau 1 – Comparaison entre les différents modes d'exécution

2.3 Agent mobile : définition, caractéristiques et propriétés :

2.3.1 Définition :

Un agent mobile est un logiciel particulier doté d'autonomie et de mobilité **[Wilhelm97]**. Cet agent se déplace à travers le réseau (en particulier l'Internet) d'un site vers un autre dans le but de fournir un service précis à son propriétaire.

2.3.2 Caractéristiques :

- **L'agent est situé** *(Situatedness)* : signifie que l'agent reçoit des entrées sensorielles à partir de son environnement et qu'il peut exécuter les actions qui changent l'environnement d'une manière quelconque.
- **L'autonomie** : signifie que l'agent peut agir sans l'intervention directe des humains (ou d'autres agents), et qu'il a le contrôle de ses propres actions et état interne.
- **La flexibilité** : peut être définie pour inclure les propriétés suivantes :
  - **Sensible** : se rapporte à la capacité d'un agent de percevoir son environnement et de répondre à temps aux modifications qui peuvent affecter l'environnement.
  - **Proactif** : les agents peuvent montrer le comportement orienté objectif, et peuvent prendre des initiatives au moment approprié.
  - **Social** : les agents devraient pouvoir agir l'un sur l'autre (avec d'autres agents et humains) afin de résoudre leurs propres problèmes et atteindre leurs objectifs, et afin d'aider d'autres avec leurs activités.

D'autres caractéristiques sont parfois discutées dans le cadre d'agent, telles que **[Jennings95]** :

- **Rationalité** : un agent ne doit pas agir de telle façon à empêcher la réalisation de ses buts, et essaie toujours de les atteindre.
- **Véracité** : le concept qu'un agent ne communiquera pas sciemment l'information fausse.
- **Bienveillance** : un agent ne peut pas avoir des buts contradictoires qui le forcent à transmettre une fausse information ou effectuer des actions qui rendent ses buts non atteints.
- **Mobilité** : la capacité d'un agent de se déplacer à travers des réseaux et entre différents serveurs pour atteindre ses buts.
- **Adaptabilité [Maes94]** : un agent doit s'adapter aux variations de son environnement pour résoudre un certain nombre de problèmes.

2.3.3 Propriétés d'agent mobile :

Le paradigme d'agent mobile à plusieurs avantages par rapport au modèle traditionnel de client/serveur **[Lange99]** :

2.3.3.1 Diminution de l'utilisation du réseau :

Le déplacement des agents mobiles permet de réduire significativement, voir supprimer, les communications distantes entre les clients et les serveurs. En privilégiant les interactions locales, l'utilisation du réseau va se limiter principalement au transfert des agents. Cette situation présente trois principaux avantages.

a) la diminution de la consommation de bande passante : En effet, plusieurs études **[Sahai98, Gray02, Bernard99]** montrent que la mise en place des agents mobiles permet d'obtenir une réduction significative de la charge réseau en termes de nombre total de données transférées. Cette diminution est constatée dans différents types d'applications nécessitant d'intenses échanges d'informations entre le client et le serveur.

b) la diminution des temps de latence **[Johansen98, Kotz99, Jun02]** : Dans le contexte des réseaux à large échelle, la mise en place d'applications reparties, nécessitant de fréquentes interactions entre client et serveur, se heurte aux temps de latence propres aux communications réseaux. Il arrive fréquemment que le temps d'attente de la réponse d'une requête soit plus long que le temps de traitement nécessaire à la réalisation du service.

c) des brèves périodes de communication : En réduisant le plus possible les communications distantes aux seuls transferts d'agents mobiles, on diminue considérablement les périodes de connexion entre deux sites.

2.3.3.2 Exécution asynchrone et autonome :

Les terminaux mobiles utilisent souvent des connections réseau chères ou fragiles. Les tâches qui nécessitent des connections continuellement ouvertes entre un terminal mobile et un réseau fixe semblent être économiquement ou techniquement pas faisables. Ces tâches peuvent être encapsulées dans des agents mobiles lancés sur le réseau sans maintenir une connexion de bout en bout. Par exemple, les PDAs (*Personal Digital Assistant*) ou les ordinateurs portatifs sont souvent déconnectés d'un réseau fixe. En outre, l'émission des

messages et les connections au réseau peuvent également engager des coûts financiers considérables. Ceci présente une occasion pour la technologie d'agent mobile parce que le terminal mobile délègue un agent pour agir en son nom et exécuter les tâches requises au lieu de l'interrogation de divers serveurs. Ces agents deviennent indépendants du processus créateur et peuvent fonctionner de façon asynchrone et autonome. Le terminal mobile se reconnecte ultérieurement pour récupérer l'agent (avec les résultats).

2.3.3.3 Adaptation dynamique :

Les agents mobiles ont la capacité de s'adapter dynamiquement aux changements de leur environnement. Ils peuvent, par exemple, réagir d'une façon autonome pour équilibrer le chargement dans le réseau ou se déplacer d'un nœud échoué à un autre. Ce comportement fournit également aux agents mobiles la robustesse et un degré de tolérance aux pannes.

2.3.3.4 Robustesse et tolérance aux fautes :

Les agents mobiles peuvent s'adapter facilement aux erreurs systèmes. Ces erreurs peuvent être physiques, par exemple la disparition d'un nœud, ou fonctionnels, par exemple l'arrêt d'un service.

Si un hôte tombe en panne, tous les agents s'exécutant sur cet hôte seront avertis, et l'agent pourra alors choisir de se déplacer vers un autre site contenant la fonctionnalité désirée. Ceci permet une bien meilleure tolérance aux fautes que le modèle statique classique.

S'il l'agent mobile vient à disparaître pour une erreur interne à son savoir faire ou pour une erreur système, il est généralement très difficile de détecter sa disparition. **[Rouvrais02]** énumère plusieurs mécanismes basés sur la réplication, transparents aux agents, permettant de résoudre en partie ce problème.

2.3.3.5 Amélioration du temps d'exécution :

L'optimisation des phases de traitement se produit à deux niveaux. Premièrement, en localisant les données et le code sur un même site, on supprime les phases de dialogue entre le client et le serveur qui sont perturbées par des temps de latence dus aux communications réseaux [**Ismael99**]. Ensuite, le déplacement du code va permettre de déléguer les calculs sur des machines serveurs (telles que des supercalculateurs) qui sont généralement plus puissantes qu'une machine cliente.

2.3.3.6 Encapsulation de protocole :

Les réseaux d'aujourd'hui se composent de beaucoup d'applications désuètes (*legacy applications*). Pendant que leurs protocoles évoluent, les problèmes de *legacy* se posent souvent. Les agents mobiles se déplacent vers l'application et encapsulent son protocole. D'autres applications communiquent avec cette application par l'intermédiaire de l'agent.

2.3.3.7 Le contrôle décentralisé :

La migration dynamique des agents simplifie la distribution des programmes centralisés pour le contrôle et la gestion des ressources du réseau et des applications distribuées **[Lange99]**.

2.4 Domaines d'application :

Vu l'utilité potentielle du paradigme d'agent mobile, la technologie d'agent mobile a réussi à pénétrer dans plusieurs domaines. Les cas d'utilisation suivants peuvent représenter des domaines où cette technologie peut faire un impact et une différence qualitative importante **[Lange99, Chess94]** :

2.4.1 Commerce électronique :

Avec l'utilisation croissante de commerce électronique le besoin d'agents mobiles augmente également. Les agents sont envoyés dans l'Internet par un utilisateur intéressé et recueillent les informations requises. Au besoin, ils peuvent collaborer avec d'autres agents **[Maes99]**. Des plans sont révisés pour créer des marchés où les agents peuvent s'occuper et même participer aux enchères. Pour cette raison un agent est équipé des mécanismes pour payer les marchandises achetées ou les services au nom de son autorité.

2.4.2 Applications de surveillance :

Un agent surveille un composant, tel qu'un pilote de périphériques, une application, ou un commutateur. Il peut réagir localement à une certaine configuration comportementale du composant surveillé. Par exemple, un agent de gestion du réseau peut surveiller le trafic de réseau jusqu'à ce qu'il détecte l'encombrement de trafic. Il envoie alors un email à l'administrateur ou prend des actions correctives appropriées.

2.4.3 Actions ambulantes :

Les transactions peuvent nécessiter plusieurs nœuds. Si les nœuds doivent être visités séquentiellement plutôt qu'en parallèle, un agent peut être une alternative à une exécution de type client/serveur.

2.4.4 Configuration de système :

Les agents peuvent fournir un mécanisme flexible pour la configuration de système. En particulier, la mobilité du code permet la reconfiguration sans la nécessité d'arrêter du système entier. Les réseaux actifs **[Tennenhouse96]**, par exemple, se fondent sur des instructions envoyées avec les paquets de transmission. La maintenance et les actualisations des systèmes dynamiques sont également supportées par ce mécanisme.

2.4.5 Traitement en parallèle :

Les agents mobiles peuvent se copier et fractionner le travail parmi les clones. Ceci permet d'exécuter des tâches en parallèle et de distribuer la puissance de traitement sur des différents nœuds. Une application basée sur la technologie d'agent mobile est ainsi plus évolutive que des applications se composant des blocs monolithiques.

2.4.6 Rassemblement de l'information :

Le rassemblement de l'information peut être réalisé par un agent mobile si des sources multiples doivent être considérées ou si les sources ne sont pas connues à l'avance. Un agent peut accumuler des connaissances pendant son itinéraire qui lui permettent de prendre une décision au sujet de son autre itinéraire.

3 SÉCURITÉ INFORMATIQUE

3.1 Définition de la sécurité informatique :

La sécurité informatique est l'ensemble des moyens mis en œuvre pour réduire la vulnérabilité d'un système contre les menaces accidentelles ou intentionnelles.

La sécurité informatique, d'une manière générale, consiste à assurer que les ressources matérielles ou logicielles d'une organisation sont uniquement utilisées dans le cadre prévu. Un ensemble des moyens techniques, organisationnels, juridiques et humains nécessaires sont mis en place pour conserver, rétablir, et garantir la sécurité de l'information et du système d'information.

La sécurité informatique vise généralement cinq principaux objectifs : **[ht2]**

- **L'intégrité :** garantit que les éléments considérés sont exacts et complets.
- **La confidentialité :** garantit que seules les personnes autorisées ont accès aux éléments considérés.
- **La disponibilité :** garantit que ces éléments considérés sont accessibles au moment voulu par les personnes autorisées.
- **L'authentification :** garantit que seules les personnes autorisées aient accès aux ressources.
- **La non-répudiation :** garantit que l'envoyeur et le destinataire sont bien les parties qui disent avoir respectivement envoyé ou reçu le message.

3.2 Politique de sécurité : **[ht3]**

Une politique de sécurité informatique est un plan d'actions définies pour maintenir un certain niveau de sécurité. Elle reflète la vision stratégique de la direction de l'organisme (industrie, administration, état, unions d'états …) en matière de sécurité informatique.

3.3 Les risques :

Tenter de sécuriser un système d'information revient à essayer de se protéger contre les risques liés à l'informatique pouvant avoir un impact sur la sécurité de celui-ci, ou des informations qu'il traite.

Le risque est définit par : $\text{risque} = (\text{menace} \times \text{vulnérabilité}) / \text{contre-mesure}$ où :

- **Les menaces :** ce sont des adversaires déterminés capables de monter une attaque exploitant une vulnérabilité.
- **Les vulnérabilités :** ce sont les failles de sécurité dans un ou plusieurs systèmes. Tout système vu dans sa globalité présente des vulnérabilités, qui peuvent être exploitables ou non.
- **Les contre-mesures :** ce sont les procédures ou techniques permettant de résoudre une vulnérabilité ou de contrer une attaque.

3.4 Les Attaques :

Les attaques représentent les moyens d'exploiter une vulnérabilité. Il peut y avoir plusieurs attaques pour une même vulnérabilité mais ça ne signifie pas que toutes les vulnérabilités sont exploitables. Les attaques informatiques sont de plus en plus motivées par des gains financiers.

Les principales attaques sont :

- Virus.
- Cheval de Troie.
- Dénis de service (*Denial of Service DoS*).
- Intrusion.
- Ecoute du réseau (*Sniffer*).

3.5 Les procédés techniques de protection :

Depuis l'extension étendue de l'informatique, beaucoup de normes, protocoles, et méthodes ont été développés dans le cadre de la sécurité d'information. Dans cette section, nous présentons quelques uns qui touchent le domaine des agents mobiles et demeurent utiles pour notre contribution.

3.5.1 Cryptographie (la science du secret) :

3.5.1.1 Vocabulaire :

- **Chiffrer** : utiliser une clé de chiffrement pour transformer (coder) un message en clair en un message incompréhensible.
- **Déchiffrer** : utiliser une clé cryptographique pour décoder un message chiffré.
- **Décrypter** : réussir à décoder un message chiffré sans avoir initialement la clé de déchiffrement.

3.5.1.2 Cryptographie symétrique :

Les algorithmes de chiffrement symétrique se fondent sur une même clé pour chiffrer et déchiffrer un message. La clé, qui doit rester totalement confidentielle, doit être transmise au correspondant de façon sûre.

Quelques algorithmes de cryptographie symétrique sont :

- AES (*Advanced Encryption Standard*)
- DES (*Data Encryption Standard*)
- Triple DES
- RC4 (*Rivest Cipher 4*)
- IDEA (*International Data Encryption Algorithm*)
- Blowfish

3.5.1.3 Cryptographie asymétrique :

La cryptographie asymétrique, ou cryptographie à clé publique est fondée sur l'existence de fonctions à sens unique. Une telle fonction est difficile à inverser, à moins de posséder une information particulière, nommée clé privée.

On diffuse à tout le monde la fonction pour coder les messages (notée clé publique) mais on garde secrète la fonction de décodage (notée clé privée).

Quelques algorithmes de cryptographie symétrique sont :

- RSA (*Rivest Shamir Adleman*)
- DSA (*Digital Signature Algorithm*)
- ElGamal
- Rabin

3.5.1.4 Protocoles courants :

- **SSL** (*Secure Socket Layer*) : SSL est un procédé de sécurisation des transactions effectuées via Internet. Le standard SSL a été mis au point par Netscape, en collaboration avec Mastercard, Bank of America, MCI et Silicon Graphics. Son principe consiste à établir un canal de communication sécurisé (chiffré) entre deux machines (un client et un serveur) après une étape d'authentification.
- **SET** (*Secure Electronic Transaction*) : SET est un protocole défini par MasterCard, VISA, et des acteurs américains intervenant sur Internet, pour sécuriser les paiements effectués par carte sur Internet. SET est basé sur l'utilisation d'une signature électronique au niveau de l'acheteur et une transaction mettant en jeu non seulement l'acheteur et le vendeur, mais aussi leurs banques respectives.
- **C-SET** (*Chip-Secure Electronic Transactions*) : C-SET est un protocole, et surtout une architecture, définis par le Groupement des Cartes Bancaires. C-SET est une adaptation du protocole SET permettant d'intégrer un environnement de sécurité physique (ce que ne fait pas SET), notamment la carte à puce, des lecteurs sécurisés de cartes à puce, et des « Boîtes Noires Commerce Electronique ».
- **S/MIME** (*Secure/Multipurpose Internet Mail Extension*) : S/MIME est un procédé de sécurisation des échanges par courrier électronique permettant de garantir la confidentialité et la non-répudiation des messages électroniques.
- **PGP** (*Pretty Good Privacy*) : PGP est une implantation du système de chiffrement RSA mise sur l'Internet par Philip Zimmerman.

3.5.2 Signature numérique :

La signature numérique permet d'authentifier l'auteur d'un document électronique et de garantir son intégrité. Un mécanisme de signature numérique doit :

- Permettre au lecteur d'un document d'identifier l'auteur qui a apposé sa signature.
- Garantir que le document n'a pas été altéré entre l'instant où l'auteur l'a signé et le moment où le lecteur le consulte.

La signature électronique n'est devenue possible qu'avec la cryptographie asymétrique. Elle se différencie de la signature écrite par le fait qu'elle n'est pas visuelle, mais correspond à une suite de nombres.

3.5.3 Certificat numérique :

Un certificat numérique (électronique) est une carte d'identité numérique pour identifier une entité physique ou non-physique. Le standard le plus utilisé pour la création des certificats numériques est le X.509.

Un certificat électronique contient les informations suivantes :

- Une clé publique.
- Le nom du propriétaire de cette clé (le propriétaire peut être une personne, une machine, un agent…).
- La durée de validité du certificat.
- D'autres informations appelées attributs.

3.5.4 Infrastructure de Gestion de Clés ICG :

L'IGC, appelée aussi PKI (*Public Key Infrastructure*) est un ensemble de composants physiques (des ordinateurs, des cartes à puces), de procédures humaines (vérifications, validation) et de logiciels (système et application) en vue de gérer le cycle de vie des certificats électroniques.

4 SÉCURITÉ POUR LES AGENTS MOBILES

4.1 Exigences de sécurité pour les agents mobiles : [Borselius02]

Les utilisateurs des systèmes distribués ont quatre exigences de sécurité principales : confidentialité, intégrité, disponibilité, et responsabilité (ou imputabilité). Les utilisateurs des agents mobiles ont également ces mêmes exigences de sécurité. Cette section fournit une brève vue d'ensemble de ces exigences.

4.1.1 Confidentialité

N'importe quelle donnée privée enregistrée sur une plate-forme ou portée par un agent doit demeurer confidentielle. Les oreilles indiscrètes peuvent recueillir des informations au sujet des activités de l'agent non seulement du contenu des messages échangés, mais également du flux de messages d'un agent à un autre agent ou à des agents différents. Un agent acheteur peut, par exemple, envoyer trois messages à un agent vendeur, suivi d'un message à sa plate-forme d'origine, et conclure la transaction avec deux messages à l'agent vendeur. L'agent vendeur peut envoyer deux messages à une agence de vérification de crédit et conclure avec un message à son agent d'opérations bancaires. Bien que le contenu des messages n'ait été jamais révélé, une oreille indiscrète peut pouvoir impliquer que les deux agents ont avec succès terminé une vente d'un certain produit.

Les agents mobiles peuvent également vouloir maintenir leur emplacement confidentiel. On doit permettre à des agents de décider si leur présence sera publiquement disponible, et les plates-formes peuvent imposer différentes politiques de sécurité sur les agents qui ont choisi d'être anonymes. Des agents mobiles qui font un shopping pour des biens et des services souhaitent le faire ainsi l'intimité, mais quand une transaction financière doit être effectuée la plate-forme peut exiger une certaine forme d'authentification.

4.1.2 Intégrité

La plate-forme d'agent doit protéger des agents contre la modification non autorisée de leur code, état, et leurs données et s'assurer que seulement les agents autorisés effectuent des modifications sur les données partagées. L'agent lui-même ne peut pas empêcher une plate-forme malveillante d'altérer son code, état, ou données, mais l'agent peut prendre des mesures pour détecter cette altération.

L'exécution sécurisée des systèmes des agents mobiles dépend également de l'intégrité des plates-formes locales et distantes. Un hôte malveillant peut facilement compromettre l'intégrité d'un agent mobile qui visite une plate-forme à distance. L'utilisateur et développeur de l'agent mobile peut ne pas avoir aucune connaissance du comportement d'une plate-forme malveillante et les effets qu'il aura sur le code de l'agent, l'état et les données qui sont sous le contrôle complet de la plate-forme. Une plate-forme malveillante peut altérer de façon subtile la séquence d'exécution du code de l'agent et engendrer des modifications difficiles à détecter aux résultats calculés. Une plate-forme malveillante peut s'ingérer dans les transactions entre les agents représentant différents organismes et falsifier les vérifications rétrospectives. Une des conséquences est que chaque organisation impliquée dans la transaction blâme l'autre de ne pas avoir fourni les biens ou les services promis.

Les attaques orientés-objectif contre les communications d'agent visent à compromettre l'intégrité des messages en changeant leur contenu, en substituant le message entier, en réutilisant un vieux message, en effaçant le message, ou en changeant la source ou la destination de message. Les attaques malveillantes sur l'intégrité des transmissions d'agent peuvent être plus graves que des erreurs de transmission résultant des voies de transmission défectueuses ou limitées.

4.1.3 Responsabilité

Chaque processus, utilisateur humain, ou agent sur une plate-forme donnée doit être jugé responsable de ses actions. Afin d'être jugé responsable, chaque agent mobile doit être identifié, authentifié, et vérifié (par un tiers de confiance).

La responsabilité exige le maintien d'un journal des événements (audit) relevant de la sécurité, et la liste de chaque événement avec l'agent (ou le processus) responsable de cet événement. Ces événements sont définis dans la politique de sécurité de la plate-forme et doivent inclure : nom d'utilisateur/agent, période d'événement, type d'événement, et succès ou échec de l'événement. Ces journaux doivent être protégés contre l'accès non autorisé et la modification.

Les audits sont également nécessaires et importants quand la plate-forme doit récupérer d'une violation de la sécurité, ou une panne matériel/logiciel. La pleine reprise d'un système défectueux exige non seulement la restauration à un état sûr et l'exécution d'un certain type de diagnostic, mais également le classement des agents avec les organismes où ils étaient affectés.

La responsabilité est également essentielle pour garantir la confiance entre les agents mobiles et les hôtes. Un agent authentifié peut se conformer à la politique de sécurité de l'hôte, mais exposer toujours un comportement malveillant en se diffusant ou en écartant délibérément l'information fausse. L'audit, bien que couteûx, peut être utilisé comme solution pour prouver la responsabilité de l'agent malveillant.

4.1.4 Disponibilité

La plate-forme d'agent doit pouvoir assurer la disponibilité des données et des services aux agents mobiles. La plate-forme doit être capable de fournir le contrôle de concurrence, le support pour l'accès simultané, la gestion de blocage, et l'accès exclusif au besoin. Les données partagées doivent être disponibles sous une forme utilisable, la capacité doit répondre aux besoins de service. La plate-forme d'agent doit être capable de détecter et récupérer les pannes du matériel et logiciel. En outre, la plate-forme peut exiger des agents d'assumer la responsabilité de leur propre faute.

La plate-forme d'agent doit pouvoir traiter les requêtes des centaines ou des milliers de visites d'agents mobiles, sinon il y aura un risque de créer un déni de service involontaire. Au cas où une plate-forme ne pourrait pas manipuler son chargement de calcul et de transmission, elle doit fournir le fonctionnement en mode dégradé des services et informer les agents mobiles qu'elle ne peut plus fournir le niveau et la qualité du service prévus.

L'assurance de la confidentialité et l'intégrité crée des restrictions sur la disponibilité et également le temps d'exécution. Le cryptage des agents et messages en transit peut imposer des retards inacceptables dans les environnements où la réponse à temps est exigée.

4.1.5 Anonymat

La plate-forme d'agent mobile doit équilibrer le besoin d'un agent d'intimité avec la nécessité de juger l'agent responsable de ses actions. La plate-forme peut rendre l'identité de l'agent secrète à d'autres agents et mettre à jour une forme d'anonymat réversible où elle peut déterminer l'identité de l'agent s'il y a besoin. Les acheteurs veulent protéger leur intimité en restant anonymes, mais les agences de crédit n'accordent pas le crédit aux consommateurs anonymes sans pouvoir vérifier leur historique et degré de solvabilité. Les problèmes d'anonymat affectant les communautés humaines se retrouvent également au niveau des communautés d'agent.

4.2 Problèmes de sécurité d'agent mobile : [Borselius02]

Dans cette section nous discuterons des problèmes de sécurité d'agent mobile basés sur les caractéristiques décrites dans la section **2.3.2**.

4.2.1 Exécution d'agent :

Les agents mobiles ont le besoin de s'exécuter quelque part. Un hôte, l'environnement immédiat d'un agent, est finalement responsable de l'exécution et la protection correctes de l'agent. Ceci mène à la question d'où des décisions de contrôle d'accès devraient être exécutées et imposées. Est-ce que l'agent comporte toutes les informations exigées ? Et si oui, est-il autorisé ?

L'environnement doit également avoir besoin d'une certaine protection contre les agents qu'il accueille. Un agent doit, par exemple, être empêché de lancer une attaque par déni de service en consommant toutes les ressources sur un hôte (serveur), et ainsi empêcher l'hôte d'effectuer autres travaux. Les problèmes de sécurité liés à l'hôte exécutant deviennent plus évidents pour les agents qui sont mobiles.

4.2.2 Autonomie :

L'autonomie peut introduire des problèmes de sécurité sérieux. L'agent peut, à titre d'exemple, être capable de réaliser des transactions et être la cible de différentes attaques qui pourraient l'empêcher de remplir ses engagements.

Il est important de mentionner aussi que les virus Internet possèdent également la propriété d'autonomie, qui leur permet de se propager efficacement sans exiger aucune interaction humaine. L'autonomie, donc, peut être utilisée pour des buts malveillants si elle n'est pas correctement contrôlée.

4.2.3 Communication :

Quelques implémentations des systèmes multi-agents MAS (*Multi Agent System*) supposent que la sécurité est fournie d'une manière transparente par une couche inférieure. Cette approche est suffisante dans un système fermé où les agents peuvent se faire confiance et le seul souci concerne des parties malveillantes externes. Cependant, les agents dans un système ouvert ont besoin d'être sensible à la sécurité (*security awareness),* c.-à-d. ils doivent pouvoir prendre des décisions basées sur d'où l'information provient et comment elle est protégé. **[He98]** suggère

que la cryptographie à clé publique et une infrastructure supportant la clé publique peuvent être utilisées comme outils importants de transmission inter-agents.

Avec la mise en place d'une infrastructure à clé publique en place, les protocoles et les mécanismes de sécurité déjà développés pour d'autres applications peuvent être amenés à se conformer aux conditions des systèmes multi-agents de fournir l'authentification, la confidentialité, et l'intégrité des données.

4.2.4 Rationalité, véracité, et bienveillance :

Ces propriétés semblent, à première vue, être nécessaires à la sécurité. Cependant, après réflexion, elles pourraient être considérées en tant que problèmes de sécurité pratiques. La signification de ces propriétés d'un point de vue de sécurité semble être : « Les agents se comportent bien et n'agiront jamais d'une façon malveillante ». Ceci n'est pas un scénario probable pour un système multi-agent qui n'est pas sous un contrôle strict, et ne correspondrait pas au scénario de système ouvert. Toutefois, des mesures peuvent être prises pour limiter le comportement malveillant des agents.

4.3 Menaces de sécurité : [Jansen00]

Les menaces de point de vue sécurité sont classifiées généralement en trois classes principales : la révélation d'information, le déni de service, et la corruption d'information. Il est important de noter que plusieurs menaces ont des contre-mesures dans les systèmes client/serveur. L'emploi des agents mobiles procure simplement une plus grande occasion à l'abus et à la mauvaise utilisation, élargissant l'échelle des menaces de manière significative.

Nous nous intéressons aux trois catégories de menaces : menaces provenant d'un agent attaquant un hôte, un hôte attaquant un agent, et un agent attaquant un autre agent sur un hôte.

4.3.1 Menaces d'agent contre hôte :

Cette catégorie représente l'ensemble de menaces dans lesquelles les agents exploitent des faiblesses de sécurité d'un hôte ou lancent des attaques contre l'hôte. Cet ensemble de menaces inclut : la mascarade, le déni de service et l'accès non autorisé.

4.3.1.1 Mascarade :

Quand un agent non autorisé utilise l'identité d'un autre agent, il est décrit une mascarade (déguisement). L'agent en déguisement peut se présenter comme étant un agent autorisé dans l'effort d'accéder aux services et aux ressources auxquels il n'a pas droit. L'agent en déguisement peut également se présenter comme étant un autre agent non autorisé pour éviter le blâme de toutes actions pour lesquelles il ne veut pas être jugé responsable. Un agent en déguisement peut endommager la confiance que l'agent légitime a établie dans une communauté d'agent et sa réputation associée.

4.3.1.2 Déni de service :

Le déni de service DoS est, d'une manière générale, l'attaque qui vise à rendre une application informatique incapable de répondre aux requêtes de ses utilisateurs.

Les agents mobiles peuvent lancer une attaque par déni de service en consommant une quantité excessive des ressources informatiques de l'hôte. Cette attaque peut être lancée intentionnellement avec des scripts en exploitant les vulnérabilités du système, ou involontairement par des erreurs de programmation. Des menaces de sécurité résultant des erreurs de programmation et des imperfections intentionnelles ont été discutées depuis le début des années 70 **[Anderson72]**.

Un agent peut comporter un code nuisible (malveillant) qui est conçu pour perturber les services offerts par l'hôte, pour dégrader l'exécution de l'hôte, ou pour extraire l'information pour laquelle il n'a aucune autorisation d'accès. Selon le niveau d'accès, l'agent pourrait être capable d'arrêter complètement l'hôte.

4.3.1.3 Accès non autorisé :

Des mécanismes de contrôle d'accès sont employés pour empêcher les utilisateurs non autorisés d'avoir accès aux services et ressources. L'application des mécanismes appropriés de contrôle d'accès exige à l'hôte d'authentifier d'abord l'identité d'un agent mobile avant qu'il soit instancié sur l'hôte. Un agent qui a l'accès à un hôte et à ses services sans avoir l'autorisation appropriée peut nuire à d'autres agents et à l'hôte lui-même. Un hôte qui accueille des agents représentant divers utilisateurs et organismes doit s'assurer que les agents n'ont pas la possibilité de lire ou d'écrire des données pour lesquelles ils n'ont aucune autorisation.

4.3.2 Menaces d'agent contre agent :

Cette catégorie représente l'ensemble de menaces dans lesquelles les agents exploitent des faiblesses de sécurité ou lancent des attaques contre d'autres agents. Cet ensemble de menaces inclut : la mascarade, le déni de service et l'accès non autorisé.

4.3.2.1 Mascarade :

La transmission inter-agent peut avoir lieu directement entre deux agents ou peut exiger la participation de l'hôte et ses services fournis. Dans ces deux cas, un agent peut essayer de déguiser son identité dans un effort de tromper l'agent avec lequel il communique. Un agent A peut se présenter, par exemple, en tant que vendeur des marchandises et de services bien connu et essaie de convaincre un autre agent B confiant de lui fournir les numéros de carte de crédit, les données de compte bancaire, une certaine forme d'argent numérique, ou d'autres informations personnelles. Le fait de se déguiser étant un autre agent nuit à l'agent B qui est trompé et l'agent dont l'identité a été assumée, particulièrement dans des sociétés d'agent où la réputation est valorisée et utilisée comme un moyen d'établir la confiance.

4.3.2.2 Déni de service :

Les agents peuvent lancer l'attaque par déni de service contre d'autres agents. Par exemple, envoyer des messages à un autre agent à plusieurs reprises peut saturer les sous-programmes de gestion de messages du destinataire. Les agents qui sont spammés peuvent choisir de bloquer des messages des agents non autorisés, mais même cette tâche exige certains traitements par l'agent ou son proxy de transmission. Si un agent est chargé par le nombre de cycles CPU qu'il consomme sur une plate-forme, spammer un agent peut causer à l'agent spammé de payer un coût monétaire en plus d'un coût d'exécution.

4.3.2.3 Accès non autorisé :

Si l'hôte n'a pas un mécanisme de contrôle, un agent peut directement gêner un autre agent en appelant ses méthodes publiques (par exemple, tentative de débordement de tampon, remise à l'état initial, etc.), ou en accédant et en modifiant les données ou le code de l'agent. La modification du code d'un agent est une forme d'attaque particulièrement insidieuse, puisqu'elle peut changer radicalement le comportement de l'agent (par exemple, transformant un agent de confiance en agent malveillant). Un agent peut également obtenir des informations sur les activités d'autres agents en utilisant des services de plate-forme pour écouter clandestinement leurs transmissions.

4.3.3 Menace de l'hôte contre agent :

Cette catégorie représente l'ensemble de menaces dans lesquelles les plates-formes compromettent la sécurité des agents. Cet ensemble de menaces inclut : la mascarade, le déni de service, l'écoute illicite, et l'altération.

4.3.3.1 Mascarade :

Un hôte peut se déguiser en tant qu'un autre hôte pour tromper un agent mobile dans sa vraie destination. Un hôte se déguisant comme un tiers de confiance peut être capable de leurrer les agents confiants et extraire des informations sensibles de ces agents.

4.3.3.2 Déni de service :

Quand un agent arrive à un hôte, il s'attend que l'hôte exécute ses demandes loyalement, fournit une répartition des ressources juste, et se conforme aux accords de qualité du service. Un hôte malveillant, cependant, peut ignorer des demandes d'agent, introduire des retards inacceptables pour des tâches critiques telles que placer des ordres de Bourse à un marché boursier, simplement ne pas exécuter le code de l'agent, ou même interrompre l'agent sans avis. Les agents sur d'autres hôtes attendant les résultats d'un agent sur un hôte malveillant doivent faire attention pour éviter le blocage total (*deadlocked*). Un agent peut également devenir activement bloqué (*livelocked*) si un hôte malveillant crée une situation dans laquelle une certaine étape critique de l'agent ne peut pas se terminer parce il a continuellement plus de travail à faire.

4.3.3.3 Écoute illicite :

La menace de l'écoute illicite classique comporte l'interception et la surveillance des transmissions secrètes. Cependant, cette menace est encore aggravée dans les systèmes des agents mobiles parce que l'hôte peut surveiller non seulement la transmission, mais aussi chaque instruction à exécuter par l'agent, toutes les données non codées ou publiques apportée, et toutes les données produites sur l'hôte. Puisque l'hôte a accès au code, à l'état, et aux données de l'agent, l'agent visiteur doit être circonspect du fait qu'il peut exposer des algorithmes de propriété industrielle, des secrets commerciaux, des stratégies de négociation, ou d'autres informations sensibles.

Quoique l'agent ne puisse pas exposer directement l'information secrète, l'hôte serait capable d'impliquer la signification des types de services demandés et de l'identité des agents communiqué. Par exemple, un agent client peut communiquer avec un agent de voyage, bien que le contenu du message ne puisse être exposé, cette transmission peut indiquer que la

personne pour qui l'agent agit, prévoit un voyage et sera absente de chez elle dans un proche avenir. L'hôte peut partager cette information impliquée avec un constructeur de valise qui peut commencer à envoyer les annonces non sollicitées, ou même pire, les administrateurs de l'hôte peuvent partager cette information avec des voleurs qui peuvent viser la demeure du voyageur.

4.3.3.4 Altération :

Puisqu'un agent peut visiter plusieurs hôtes sous divers domaines de sécurité durant sa vie, les mécanismes doivent être en place pour assurer l'intégrité du code, état et données de l'agent. Un hôte malveillant doit être empêché de modifier le code, l'état, ou les données d'un agent sans être détecté. La détection des modifications malveillantes sur l'état d'un agent pendant son exécution ou les données produites lors de sa visite d'un hôte malveillant n'a pas encore trouvé de solution définitive.

4.4 Contre-mesures :

Dans cette section, nous présentons quelques approches adressant les problèmes de sécurité pour le paradigme agent mobile. Nous distinguons deux catégories : d'une part, la protection des hôtes contre des agents mobiles malveillants, et d'autre part, la protection des agents vis-à-vis des hôtes malveillants. Les deux volets sont bien détaillés dans **[Hacini08]**.

4.4.1 Protection des agents mobile :

4.4.1.1 Matériel de confiance :

Wilhelm **[Wilhelm98]** propose une approche de ce type basée sur ce qu'il appelle « environnement d'exécution de confiance » (*Trusted Processing Environment TPE*). Dans cette approche, les agents sont encryptés par la machine de leur propriétaire et ne seront plus exécutés par la plate-forme visitée mais par le périphérique TPE.

Afin de généraliser l'utilisation du concept de périphérique spécial, Mana **[Mana02]** a proposé l'utilisation de carte à puces. Son idée consiste à subdiviser le code de l'agent en sections dont certaines seront encryptées par la clé publique d'une carte à puces. Les sections cryptées seront transmises à la carte pour être décryptées et exécutées à l'intérieur de cette carte. La paire de clés est générée à l'intérieur de la carte, la clé publique sera publiée tandis que la clé secrète ne sera jamais révélée **[Mana02]**.

4.4.1.2 Nœuds de confiance :

Cette approche est basée sur l'installation d'un ensemble d'hôtes de confiance. Ceci peut être fait en encryptant l'agent mobile durant son cheminement et en authentifiant l'hôte avant que l'agent mobile ne lui soit transmis.

En introduisant des nœuds de confiance au sein d'une infrastructure, vers laquelle les agents peuvent, si nécessaire, migrer, il est possible d'éviter que des informations sensibles ne soient émises vers des hôtes non fiables.

Cette approche va à l'encontre de la notion d'agent mobile. Si un agent mobile a un itinéraire préétabli, l'avantage de la vaste quantité de ressources disponibles sur l'Internet peut être perdu ou obstrué.

4.4.1.3 Obscurcissement :

L'obscurcissement consiste à transformer un programme en un autre programme équivalent mais difficile à comprendre. Ces transformations sont évaluées selon leur puissance de confusion, leur résilience (la capacité de résister aux attaques de « dé-obscurcissement »), et leur coût **[Collberg97]**.

Une approche logicielle a été proposée par Hohl **[Hohl98]** pour assurer la sécurisation de l'agent par son propre code. L'approche consiste à convertir un agent A en un agent B (boite noire) en utilisant des algorithmes d'obscurcissement. L'agent B est analogue au premier agent A sur le plan fonctionnalités mais son code est difficile à analyser **[Wilhem97]**.

Puisque cette technique ne permet pas la protection absolue de code **[Barak01, D'Anna03]**, les données et le code d'agent sont protégées temporairement durant une certaine période de validité.

Pour produire un code illisible, Hohl propose la violation totale des règles du génie logiciel telles que : l'attribution des noms significatifs aux variables ou l'écriture d'un code modulaire. Il est possible d'insérer des fragments de code mort pour améliorer la protection. Seulement il faut se méfier des mécanismes de détection de codes morts.

L'approche d'obscurcissement est une solution logicielle efficace pour la sécurisation des agents. Toutefois, elle a des limites théoriques et techniques **[Barak01]**.

4.4.1.4 Trace d'exécution :

Cette approche **[Vigna97, Vigna98]** a été proposée pour vérifier l'intégrité des agents après leurs retours et détecter tout comportement malicieux tel que la modification du code de l'agent mobile ou de son état. Cette technique exige la création et la maintenance d'une trace non répudiable (signée par la plate-forme visitée) comportant les opérations effectuées par l'agent au cours de son exécution.

Le processus de cette technique est comme suit :

La plate-forme A reçoit un agent de sa plate-forme d'origine. Après l'avoir exécuté, A envoie un message signé contenant le code de l'agent p, son état SA, et sa trace d'exécution TA vers la prochaine plate-forme B. B sauvegarde ce message et envoie ensuite un accusé de réception signé à A. B exécute ensuite l'agent, ce qui résulte un nouvel état SB et une nouvelle trace TB. B signe ensuite le tout (p, SB, TB) et l'accompagne avec l'agent lors de sa transmission vers la plate-forme suivante. Ce processus continue jusqu'à ce que l'agent revienne à sa plate-forme d'origine.

Après retour de l'agent, si le propriétaire a un doute envers une plate-forme donnée, il pourra simuler l'exécution sur la plate-forme soupçonnée et comparer la simulation avec la trace reçue.

Cette approche assure la non-répudiation et la détection de toute manipulation de l'agent. Toutefois, elle reste détective et n'empêche pas l'utilisation frauduleuse d'agent. De plus, la taille potentiellement grande de la trace et le nombre de messages échangés rendent cette technique assez coûteuse. Cette approche a été améliorée par Hohl **[Hohl00]**, et ensuite par Tan et Moreau **[Tan01a, Tan02b]** pour pallier certains de ces problèmes.

4.4.1.5 Calcul par fonction cryptographique :

Cette approche a été proposée par Sander et Tschudin **[Sander98a, Sander98b]**. Elle est basée sur l'exécution, sur une plate-forme d'agent, d'un programme englobant une fonction encryptée.

Le problème est énoncé comme suit :

Alice possède un algorithme qui calcule une fonction *f*. Bob possède une données *x* et désire calculer *f(x)* pour elle, mais Alice ne veut pas que Bob connaisse *f*. De plus, Bob n'a pas besoin d'interagir avec Alice durant le calcul.

L'idée est la suivante :

(1) Alice encrypte *f* (une fonction matérialisée par un agent A) en *E(f)*. (2) Alice écrit un programme *P(E(f))* pour implémenter *E(f)*, ce qui produit un nouvel agent B. (3) Alice envoie *P(E(f))* à Bob (migration de l'agent). (4) Bob exécute l'agent B *P(E(f))* sur la donnée *x*. (5) Bob envoie *P(E(f))(x)* à Alice. (6) Alice décrypte *P(E(f))(x)* en utilisant l'algorithme de décryptage E^{-1}.

Cette approche assure la confidentialité, mais n'empêche pas la réexécution, l'extraction de code, ou le déni de service. Toutefois, le codage est appliqué uniquement à l'ensemble des fonctions polynomiales et rationnelles **[Sander98b]**. En plus, l'exécution de l'agent se limite au calcul d'un résultat sur une plate-forme distante, puis l'agent retourne à sa plate-forme d'origine, ce qui empêche la négociation et les prises de décision.

4.4.1.6 Clé environnementale :

Cette approche proposée par **[Riordan98]** permet à l'agent mobile d'exécuter des actions prédéfinies quand seulement certaines conditions sont vérifiées au niveau de sa plate-forme. La technique de la généralisation de la clé environnementale peut être lorsqu'une plate-forme désire communiquer avec une autre en lui envoyant un message (transporté par un agent mobile) chiffré incluant des données et/ou du code exécutable. L'agent mobile attendra au niveau de la plate-forme de réception jusqu'à ce qu'un certain état environnemental se produise pour générer une clé d'activation et déchiffrer ensuite le message transporté.

Un inconvénient de cette approche est qu'une plate-forme qui contrôle complètement l'agent pourrait simplement modifier l'agent pour imprimer le code à la réception, au lieu de l'exécuter **[Jansen00]**.

4.4.1.7 Appréciation d'état :

Cette approche, définit par Farmer et al. **[Farmer96]**, identifie la notion d'évaluation d'état, qui est un mécanisme permettant à un agent d'évaluer les privilèges dont il dispose sur un site (plate-forme) particulier. Ce qui permet de limiter les actions de l'agent par son propriétaire sur chaque plate-forme visitée.

L'approche repose sur une fonction protégée permettant, à partir d'un ensemble de variables d'état, de vérifier la stabilité de l'état d'un agent mobile. Si la fonction d'évaluation d'état ne demande pas le privilège « exécuter », puisque l'agent est dans un état peu sûr, la plate-forme ne l'exécutera pas, et limite ainsi les dommages qui peuvent être causés par l'agent altéré.

Jansen **[Jansen01a, Jansen01b, Jansen01c]** améliore cette approche et propose la séparation entre la structure de données de l'agent et son propre code. La structure est définit dans un certificat X.509. L'approche protège l'agent contre une utilisation illicite en fournissant une preuve des intentions réelles de son producteur/émetteur, mais n'offre aucune protection des données que l'agent transporte.

Sayeb **[Sayeb02]** propose une autre approche, qui consiste à utiliser un arbre de diagnostic permettant la détection d'une attaque possible par l'association d'un symptôme à chaque attaque et la définition d'un arbre de combinaison de ces symptômes.

L'appréciation d'état permet la détection automatique des manipulations qui mettent un agent mobile dans un état inacceptable. Cette détection est conditionnée par l'exécution réelle des fonctions d'évaluation, ce qui est le cas uniquement d'une plate-forme d'agent honnête.

4.4.1.8 Agents coopérants :

Cette approche a été proposée par Roth **[Roth99]** dans l'objectif d'assurer la protection en faisant partager la tâche par plusieurs agents coopérants. Il propose, dans un exemple explicatif, de définir deux sous-groupes indépendants de plates-formes que l'agent visitera. Un des deux groupes est composé uniquement d'hôtes fiables. Basé sur le clonage, deux agents seront crées pour partager une tâche, le premier agent migre uniquement vers les plates-formes du premier groupe tandis que le second agent migre uniquement vers les plates-formes appartenant au second groupe. Avant de migrer, un agent envoie à l'autre agent l'identité de sa plate-forme actuelle, ainsi que celle de la plate-forme où il se rend. L'autre agent maintient un journal d'itinéraire et vérifie qu'il n y'a pas de contradiction.

Dans cette technique, les tâches critiques d'un seul agent mobile peuvent être distribuées entre des agents coopératifs. Elle réduit la possibilité de vol des données partagées, mais son inconvénient majeur est l'abus d'utilisation des ressources réseau pour assurer la communication entre les agents coopérants. Enfin elle protège uniquement les données et non l'agent en totalité.

4.4.1.9 Fonction de validation :

Blum **[Blum88]** propose les approches à base de fonction de validation pour vérifier la fiabilité d'un code. Ces approches peuvent être appliquées dans le but de vérifier l'intégrité d'un agent mobile sur une plate-forme distante. Une approche plus récente proposée par Loureiro **[Loureiro01a]** définit une fonction de validation V comme suit : Soient P un programme qui implémente une fonction *f*, Y l'ensemble des résultats possibles de $f(x_i)$/ xi $\in$ $\{0,1\}^n$ et D(y') le résultat reçu après exécution à distance. La fonction V satisfait la condition :

Si (y = D(y') $\notin$ Y) alors P(V(y') = accept) < δ, où P représente la probabilité et δ la probabilité d'erreur.

4.4.2 Protection des hôtes :

4.4.2.1 Bac à sable (Sandboxing) :

La technique du bac à sable permet à des programmes non fiables d'être exécutés dans leur espace virtuel en les empêchant, de ce fait, d'intervenir avec d'autres applications.

Cette technique **[Vigna98]** consiste à exécuter le code de l'agent mobile dans un environnement restreint (*sandbox*) qui apparaît comme étant le système dans sa globalité **[Gong98]**. Les agents ont des privilèges limités et sont interprétés de manière sécurisé. Un exemple de cette technique est l'exécution des applets Java à l'intérieur d'un navigateur Web.

Un inconvénient de cette technique est qu'elle augmente le temps d'exécution d'un code distant légitime **[Wahbe93]**.

4.4.2.2 Signature de code :

L'agent mobile est signé numériquement par son auteur afin qu'il puisse être identifié durant ses déplacements. Cette technique assure une authentification de haut niveau pour les hôtes, et l'intégrité du code pour l'hôte visité **[Reiser00]**. La signature de code d'agent mobile confirme son origine, intégrité et autorité.

Cette technique ne résolut pas les risques liés au comportement malicieux d'un agent provenant d'une source fiable.

4.4.2.3 Historique de l'itinéraire :

L'idée de cette approche est de maintenir un enregistrement authentifiable des plates-formes déjà visitées par un agent **[Reiser00]**, et permet, ainsi, à une plate-forme de connaitre où l'agent était exécuté avant. La nouvelle plate-forme se base sur l'historique de l'itinéraire (*path history*) pour décider de l'exécution ou non de l'agent et des privilèges qu'elle doit lui accorder. L'historique de l'itinéraire exige de chaque plate-forme de rajouter une entrée signée indiquant son identité et l'identité de la prochaine plate-forme à visiter.

L'inconvénient majeur de cette approche est que le contrôle de l'itinéraire devient de plus en plus coûteux au fur et à mesure que celui-ci grandit **[Jansen00]**.

4.4.2.4 Evaluation d'état :

L'objectif d'évaluation d'état (*state appraisal*) **[Acharya97]** est d'assurer que l'état d'un agent n'a pas été trafiqué et que l'agent n'effectuera aucune action illégale par une fonction d'évaluation d'état qui devient une partie du code d'agent. L'auteur d'agent fournit les fonctions d'évaluation signées. Une plate-forme d'agent utilise ces fonctions pour vérifier l'état d'un agent arrivé et déterminer les privilèges accordés pendant l'exécution.

Le problème principal des cette technique est la difficulté de formuler les propriétés appropriées de sécurité pour l'agent mobile, et d'obtenir une fonction d'évaluation d'état qui garantit ces propriétés **[Swarup97].**

4.4.2.5 Vérification de code :

L'approche de vérification de code (*Proof Carrying Code PCC*) **[Necula98]** exige que lors de la mise en route de l'agent, son créateur doit montrer formellement que l'agent se conforme à une certaine politique de sécurité en fournissant un ensemble de preuves intégrées et transportées par l'agent mobile. La nouvelle hôte visitée n'exécute le code que seulement si la preuve est correcte.

L'inconvénient majeur de cette approche est la difficulté de produire de telles preuves formelles d'une manière automatisée et efficace, et jusqu'à l'heure actuelle, les preuves doivent généralement être produites de façon manuelle **[Loureiro01b]**.

5 CONCLUSION

La maîtrise technologique des agents mobiles est un enjeu économique et stratégique très important, ce qui explique la difficulté de trouver une normalisation acceptable. Les problèmes de sécurité pour les agents statiques peuvent, au moins dans la théorie, être abordés par la technologie de sécurité et les protocoles existants. Par contre, la sécurité des agents mobiles est le problème le plus important qui empêche leur adoption actuelle. En effet, même si la protection des plates-formes est quasiment assurée, celle des agents reste un réel problème qui n'a pas encore trouvé de solution définitive. Les problèmes de sécurité liés à l'utilisation des agents mobiles constituent donc un champ d'investigation complet.

Nous avons présenté, dans ce chapitre, la notion de mobilité, les différentes caractéristiques et propriétés des agents mobiles, et leurs domaines d'application. Nous avons ensuite présenté, brièvement, la notion de la sécurité informatique, les différents concepts liés, et quelques procédés de protection, à savoir, la cryptographie, la signature numérique, et les protocoles. Finalement nous avons expliqué les exigences et problèmes de sécurité pour les agents mobiles, et présenté les différentes approches relatives à la protection des agents mobiles.

L'exécution des agents mobiles sur les plates-formes non fiables est un facteur qui introduit des problèmes sérieux de sécurité, en particulier liés à l'exécution correcte de l'agent et à la confidentialité de ses données. Différents normes et modèles ont été développés afin de pallier ces problèmes, par conséquence, l'utilisation des agents mobiles dans un environnement distribué et dynamique crée, dans un contexte de sécurité, un problème d'hétérogénéité et interopérabilité. Ce problème donne aux ontologies une grande importance comme étant une solution efficace pour éliminer l'hétérogénéité et réussir l'interopérabilité. Le domaine d'ingénierie ontologique constitue l'objet de notre prochain chapitre.

CHAPITRE 2

L'INGÉNIERIE ONTOLOGIQUE

« Entre ce que je pense,
ce que je veux dire,
ce que je crois dire,
ce que je dis,
ce que vous voulez entendre,
ce que vous entendez,
ce que vous croyez en comprendre,
ce que vous voulez comprendre, et ce que vous comprenez,
il y a au moins neuf possibilités de ne pas se comprendre. Mais essayons quand même... » **- Bernard Werber.**

1 INTRODUCTION

Nées des besoins de représentation des connaissances, les ontologies sont à l'heure actuelle au cœur des travaux menés en Ingénierie des Connaissances (IC). Le terme « ontologie » est utilisé depuis le début des années 1990, et son champ d'application s'élargit considérablement. Un des plus grands projets basés sur l'utilisation d'ontologies consiste à ajouter au Web une véritable couche de connaissances permettant des recherches d'informations au niveau sémantique. A terme, il est prévu que des applications internet pourront mener des raisonnements utilisant les connaissances stockées sur la Toile.

Au fur et à mesure des expérimentations, des méthodologies de construction d'ontologies et des outils de développement adéquats sont apparus. L'enjeu de l'effort engagé est de rendre les machines suffisamment sophistiquées pour qu'elles puissent intégrer le sens des informations.

Dans ce chapitre, nous relèverons les différentes définitions qui ont été attribuées à la notion d'ontologie. Nous montrerons aussi le but de l'utilisation des ontologies dans le domaine du Web sémantique ainsi que leur place dans les systèmes à bases de connaissances. Ensuite, nous montrerons le processus et les méthodologies servant leur construction, ainsi que les principaux formalismes et les outils utilisés pour servir leur représentation et développement, à savoir les langages de spécification, les moteurs d'inférences, les langages d'interrogation, et les éditeurs d'ontologie. Finalement, nous présenterons quelques approches utilisant l'ontologie dans le domaine de sécurité.

2 NOTION D'ONTOLOGIE

Introduit en Intelligence Artificielle (IA) il y a 23 ans, le terme d'ontologie est cependant usité en philosophie depuis le XIX^{ème} siècle. Dans ce domaine, L'ontologie est une étude de l'être en tant qu'être, c'est-à-dire, une étude des propriétés générales de ce qui existe. C'est à l'occasion de l'émergence de l'Ingénierie des Connaissances que les ontologies sont apparues en IA, comme réponses aux problématiques de représentation et de manipulation des connaissances au sein des systèmes informatiques.

2.1 Définitions

Le terme « ontologie » est employé dans des contextes très différents touchant la philosophie, la linguistique ou l'IA. De nombreuses définitions ont été offertes pour donner un éclaircissement sur ce terme, mais aucune de ces définitions ne s'est explicitement imposée. Les définitions de ce terme ne sont pas toujours consistantes et cela dépend des domaines spécifiques **[Noy01]**. Pour ne pas dévier de notre propos, nous avons recensé les définitions suivantes :

Neches et al. **[Neches91]** furent les premiers à proposer une définition :

Définition1 « *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* » **[Neches91]**.

Définition2 « *une ontologie est une spécification explicite d'une conceptualisation* » **[Gruber93].**

La définition de Gruber est la plus utilisée dans la littérature. Elle a été légèrement modifiée par Borst :

Définition3 *« une ontologie est une spécification explicite et formelle d'une conceptualisation partagée »* **[Borst97]**.

Le terme « *conceptualisation* » réfère dans cette définition à une abstraction d'un phénomène du monde, obtenue en identifiant les concepts appropriés à ce phénomène. Le terme « *formelle* » indique que les ontologies sont interprétables par la machine. Cependant, « *spécification explicite* » signifie que les concepts de l'ontologie et les contraintes liées à leur usage sont définis de façon déclarative. Enfin, le terme «*partagé*» signifie que l'ontologie capture la connaissance consensuelle. Mais cette définition laisse la porte ouverte à de nombreuses définitions.

Définition4 *« une ontologie est une description formelle d'entités et leurs propriétés, relations, contraintes, comportement »* **[Grüninger95]**

Cette définition de Grüninger est simplifiée dans **[Ikeda97]** où une ontologie est définie comme un ensemble de définitions de concepts et leurs relations. A ne pas confondre avec un modèle qui est un ensemble d'instances de ces concepts.

Définition5 *« Les ontologies sont des spécifications partielles et formelles d'une conceptualisation commune »* **[Guarino97].**

En 1997, Guarino accentue l'ambigüité du terme conceptualisation qui doit être pris dans son sens intuitif. La spécification des ontologies est partielle, car une conceptualisation ne peut pas toujours être entièrement formalisée dans un cadre logique, du fait d'ambiguïtés ou du fait qu'aucune représentation de leur sémantique n'existe dans le langage de représentation d'ontologies choisi. « *Commune* » renvoie à l'idée qu'une ontologie rend compte d'un savoir consensuel, c'est-à-dire qu'elle n'est pas l'objet d'un individu, mais qu'elle est reconnue par un groupe **[Furst02]**.

2.2 Que représente-t-on dans une ontologie ?

Les ontologies produisent un vocabulaire commun d'un domaine et définissent, de façon plus ou moins formelle, la signification des termes et des relations entre eux. Les connaissances intégrées dans les ontologies sont formalisées en mettant en jeu cinq types de composants : concepts, relations, fonctions, axiomes, instances **[Gomez-Pérez99]**.

Concepts : Ils sont appelés aussi termes ou classes de l'ontologie. Un concept est un constituant de la pensée (un principe, une idée, une notion abstraite) sémantiquement évaluable et communicable. L'ensemble des propriétés d'un concept constitue sa compréhension ou son intension et l'ensemble des êtres qu'il englobe, son extension. Selon **[Bachimont04]** un concept se définit à trois niveaux : Un concept est une signification. Sa place dans un système de significations permet de le comprendre, de le distinguer et de le différencier par rapport à d'autres concepts. Un concept est une construction.

Selon **[Gomez-Pérez99]**, ces concepts peuvent être classifiés selon plusieurs dimensions :

- Niveau d'abstraction (concret ou abstrait).
- Atomicité (élémentaire ou composée).

- Niveau de réalité (réel ou fictif).

En résumé, un concept peut être tout ce qui peut être évoqué et peut consister en la description d'une tâche, d'une fonction, d'une action, d'une stratégie ou d'un processus de raisonnement, etc.

Relations : Représentent un type d'interaction, ou bien des associations existant entre les concepts d'un domaine. Elles se définissent formellement à partir d'un produit de n concepts :

$R : C_1 \times C_2 \times \ldots \times C_{n-1} \rightarrow C_n$

sous-classe-de (Spécialisation, généralisation), partie-de (agrégation ou composition), associée-à, instance-de sont des exemples de relations binaires.

Voici quelques relations les plus courantes dans la littérature :

1) L'équivalence : une relation R est une relation d'équivalence si et seulement si : R est symétrique, réflexive et transitive. On écrit :

(R est une relation d'équivalence) ⇔ *((R symétrique) ^ (R réflexive) ^ (R transitive))*

2) la cardinalité : c'est le nombre possible de relations de ce type entre les mêmes concepts (ou instances de concept). Les relations portant une cardinalité représentent souvent des attributs. Exemple : une pièce a au moins une porte, un humain a entre zéro et deux jambes.

3) L'incompatibilité : Deux relations sont incompatibles si elles ne peuvent lier les mêmes instances de concepts. Exemple : les relations «être rouge » et «être vert » sont incompatibles ;

4) L'inverse : Deux relations binaires sont inverses l'une de l'autre si, quand l'une lie deux instances I1 et I2, l'autre lie I2 et I1. Exemple : les relations « a pour père » et « a pour enfant » sont inverses l'une de l'autre ;

5) L'exclusivité : Deux relations sont exclusives si, quand l'une lie des instances de concepts, l'autre ne lie pas ces instances, et vice-versa. L'exclusivité entraîne l'incompatibilité. Exemple : l'appartenance et la non appartenance sont exclusives.

Et bien d'autres relations…

Fonctions : ce sont des cas particuliers de relations dans lesquelles le $N^{ième}$ élément de la relation est défini de manière unique à partir des n-1 premiers. Formellement, les fonctions sont définies ainsi : $F: C_1 \times C_2 \times \ldots \times C_{n-1} \rightarrow C_n$.

Comme exemple de fonctions binaires, nous avons la fonction mère de et le carré, et comme exemple de fonction ternaire, le prix d'une voiture usagée sur lequel on peut se baser pour calculer le prix d'une voiture d'occasion en fonction de son modèle, de sa date de construction et de son kilométrage.

Axiomes : constituent des assertions, acceptées comme vraies, à propos des abstractions du domaine, traduites par l'ontologie. Ils ont pour objectif de représenter des concepts et des relations dans un langage logique permettant de représenter leur sémantique. Ils représentent les intentions des concepts et des relations du domaine et, de manière générale, les connaissances n'ayant pas un caractère strictement terminologique **[Staab00]**. L'utilisation des axiomes sert à définir le sens des entités, mettre des restrictions sur la valeur des attributs, examiner la conformité des informations spécifiées ou en déduire de nouvelles.

Instances: elles constituent la définition extensionnelle de l'ontologie; ces objets véhiculent les connaissances (statiques, factuelles) à propos du domaine du problème.

2.3 Différentes sortes d'ontologies

Cette section n'a pas l'ambition de fournir une typologie approfondie des ontologies. Cependant, elle présente les types d'ontologies les plus généralement utilisés. Nous pouvons classer les ontologies selon plusieurs dimensions. Parmi celles-ci, nous en examinerons deux *1) Objet de conceptualisation ; 2) Niveau de formalisme de représentation*

2.3.1 Objet de conceptualisation

Dans **[Gomez-Pérez99],** Les ontologies sont classifiées selon leur objet de conceptualisation (le but de leur utilisation) de la façon suivante : *1) Supérieure/Haut niveau, 2) Domaine, 3) Tâche, 4) Application.*

Ontologie de haut niveau : décrit des concepts très généraux comme l'espace, le temps, la matière, les objets, les événements, les actions, etc. Ces concepts ne dépendent pas d'un problème ou d'un domaine particulier, et doivent être, du moins en théorie, consensuels à de grandes communautés d'utilisateurs **[Guarino98]**.

Ontologie de domaine : Contrairement aux ontologies de haut niveau, les ontologies de domaine sont plus spécifiques. Elles synthétisent les connaissances spécifiques à un domaine particulier. Elles décrivent le vocabulaire ayant trait à un domaine générique (ex. : l'enseignement, la médecine…), notamment en spécialisant les concepts d'une ontologie de haut niveau **[Guarino98]**.

Ontologie de tâches : Ce type d'ontologies est utilisé pour conceptualiser des tâches spécifiques dans les systèmes, telles que les tâches de diagnostic, de planification, de conception, de configuration, de tutorat. Soit tout ce qui concerne la résolution de problèmes. Ce type d'ontologies décrit le vocabulaire concernant une tâche générique (ex. : enseigner, diagnostiquer…), notamment en spécialisant les concepts d'une ontologie de haut niveau **[Guarino98]**.

Ontologie d'application: Cette ontologie est la plus spécifique, elle contient des concepts dépendants d'un domaine et d'une tâche particuliers, qui sont généralement subsumés par des concepts de ces deux ontologies. Ces concepts correspondent souvent aux rôles joués par les entités du domaine lors de l'exécution d'une certaine activité **[Guarino98]**.

2.3.2 Niveau de formalisme de représentation

Selon le niveau du formalisme de représentation, **[Uschold96]** propos une classification comprenant quatre catégories :

1. **Informelles** : ontologies opérationnelles dans un langage naturel (sémantique ouverte).
2. **Semi-informelles** : utilisation d'un langage naturel structuré et limité.
3. **Semi-formelles** : langage artificiel défini formellement.
4. **Formelles** : utilisation d'un langage artificiel contenant une sémantique formelle, ainsi que des théorèmes et des preuves de propriétés telles la robustesse et l'exhaustivité **[Gomez-Pérez99]**.

2.4 Les ontologies et les systèmes à bases de connaissances

Les ontologies sont apparues en informatique, plus précisément en Ingénierie des Connaissances, dans le cadre des démarches d'acquisition des connaissances pour les systèmes à base de connaissances (SBC). Les SBC proposaient alors de spécifier, d'un coté des connaissances du domaine modélisé, et de l'autre, des connaissances de raisonnement qui manipule et utilise ces connaissances du domaine. L'idée de cette séparation modulaire était de construire mieux et plus rapidement des SBC en réutilisant le plus possible des composants génériques, que ce soit au niveau du raisonnement ou des connaissances du domaine **[Bachimont03]**.

En conclusion, l'objectif de l'ingénierie ontologique est de diversifier les applications des Systèmes à Base de Connaissances (SBC), et de permettre une représentation des connaissances indépendantes de ces diverses applications, de manière à assurer sa portabilité d'une application à l'autre.

2.5 Les ontologies et le web sémantique

Le Web actuel est essentiellement syntaxique, la structure des ressources étant bien définie, mais leur contenu restant inaccessible aux traitements machines, seuls les humains étant capables de l'interpréter.

Le Web sémantique **[Lee02]** a alors l'ambition de lever cette difficulté en associant aux ressources du Web des entités ontologiques comme références sémantiques, ce qui permettra aux différents agents logiciels d'accéder et d'exploiter directement le contenu des ressources et de raisonner dessus. Ce référencement sémantique peut aussi résoudre les problèmes d'interprétation des ressources informationnelles provenant des applications hétérogènes et réparties et de permettre ainsi à ces applications d'être intégrées sémantiquement **[Uschold02]**.

L'architecture du Web sémantique repose sur une hiérarchie des langages d'assertion et de description d'ontologies ainsi que sur un ensemble de services pour l'accès aux ressources au moyen de leurs références sémantiques, pour gérer l'évolution des ontologies, pour l'utilisation des moteurs d'inférences capables d'effectuer des raisonnements complexes ainsi que des services pour la vérification de la validité sémantique de ces raisonnements **[Oberle04]**.

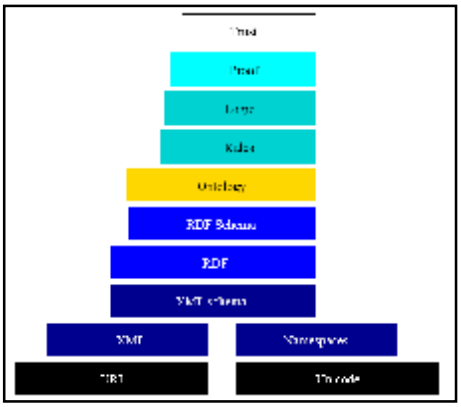

Figure 4 – Les couches du Web sémantique

3 LES ONTOLOGIES : DIFFERENTS BESOINS

Les ontologies sont utilisées dans plusieurs domaines, les plus répandus sont :

- Communication.
- Interopérabilité entre les systèmes.
- Ingénierie des systèmes.

La figure ci-dessous montre les domaines d'utilisations des ontologies

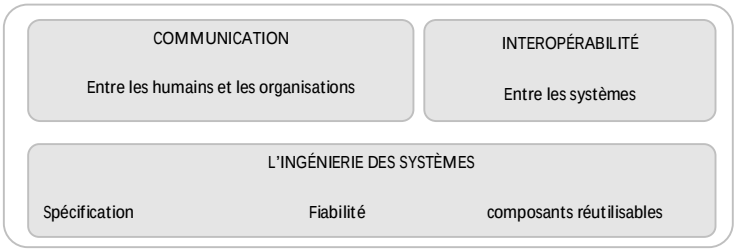

Figure 5 – Domaines d'utilisation des Ontologies.

3.1 Communication :

Les humains peuvent communiquer efficacement s'ils ont des connaissances ou des points de vue partagés. Ces connaissances partagées peuvent être obtenues si le domaine est explicitement décrit sans confusion terminologique ou conceptuelle pour être compris de la même façon par tout le monde.

Une ontologie facilite la communication en fournissant une spécification explicite d'un domaine qui représente un modèle normatif. De plus, les ontologies permettent d'assurer la consistante et d'enlever l'ambiguïté dans les descriptions des connaissances concernant un domaine spécifique. Finalement, les ontologies peuvent intégrer différentes perspectives des utilisateurs. Quand les utilisateurs (qui ont différentes perspectives d'un domaine) partagent une ontologie, ils ont une perspective standard.

3.2 Interopérabilité :

L'interopérabilité implique la possibilité de pouvoir demander et recevoir des services entre des systèmes interopérables. Deux systèmes sont considérés interopérables s'ils vérifient les deux conditions suivantes :

- Ils opèrent comme une unité afin de réaliser une tâche commune.
- Ils peuvent échanger des messages et des requêtes.

Les ontologies permettent de faciliter l'interopérabilité en intégrant les connaissances concernant différents domaines dont l'objectif est de décrire un domaine unifié ou accomplir une tâche commune. Elles permettent aussi d'intégrer les différents vocabulaires concernant certains domaines. Pour ce faire, les ontologies de ces domaines doivent être intégrées par les méthodes d'intégration d'ontologies afin de partager un même vocabulaire.

3.3 Ingénierie des systèmes :

Le développement des systèmes basé sur les ontologies a donné un profit à l'ingénierie de systèmes qui peut être résumé comme suit:

- **Réutilisabilité** : l'ontologie encode les informations relatives à un domaine (y compris les composants logiciels) de sorte que le partage et la réutilisation sont possibles.
- **Acquisition des connaissances** : l'ontologie guide l'acquisition des connaissances.
- **Sûreté** : l'ontologie rend possible l'automatisation du processus de vérification de consistance.
- **Spécification** : l'ontologie aide le processus d'identification des besoins et la définition des spécifications des systèmes.

4 UN SQUELETTE DE METHODOLOGIE POUR CONSTRUIRE DES ONTOLOGIES

Le processus de construction d'une ontologie est une collaboration qui réunit des experts du domaine de connaissance, des ingénieurs de la connaissance, voire les futurs utilisateurs de l'ontologie. Cette collaboration ne peut être fructueuse que si les objectifs du processus ont été clairement définis, ainsi que les besoins qui en découlent. La Figure 6 représente le processus de construction d'ontologie.

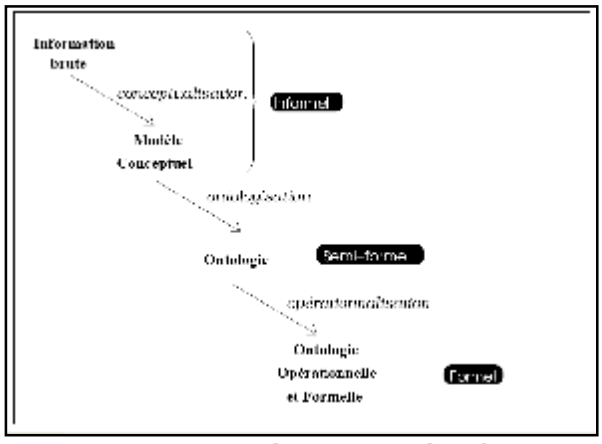

Figure 6 – Processus de construction d'ontologie

4.1 Evaluation des besoins

Le but visé par la construction d'une ontologie se décline en 3 aspects:

L'objectif opérationnel : il est indispensable de bien préciser l'objectif opérationnel de l'ontologie, en particulier à travers des scénarios d'usage.

Le domaine de connaissance : il doit être délimité aussi précisément que possible.

Les utilisateurs : ils doivent être identifiés autant que faire se peut, ce qui permet de choisir, en accord avec l'objectif opérationnel, le degré de formalisme de l'ontologie, et sa granularité.

Une fois le but défini, le processus de construction de l'ontologie peut démarrer, en commençant par la phase de conceptualisation.

4.2 Conceptualisation :

Cette étape permet d'aboutir à un modèle informel, donc sémantiquement ambiguë et généralement exprimé en langage naturel. Elle consiste, à partir des données brutes, à dégager les concepts et les relations entre ces concepts permettant de décrire de manière informelle les entités cognitives du domaine.

L'objectif est d'aboutir à un modèle conceptuel, ce modèle consiste en un ensemble de termes désignant les entités du domaine de connaissances (concepts, relations, propriétés des concepts et des relations, …), assortis d'informations exprimant leur sémantique. La découverte des connaissances d'un domaine peut s'appuyer à la fois sur l'analyse de documents et sur l'interview d'experts du domaine. Ces activités doivent être raffinées au fur et à mesure que la conceptualisation émerge.

4.3 Ontologisation :

L'ontologisation consiste en une formalisation partielle, sans perte d'information, du modèle conceptuel obtenu dans l'étape précédente. Ce qui permet de faciliter sa représentation ultérieure dans un langage complètement formel et opérationnel.

Elle effectue une transcription des connaissances dans un certain formalisme de connaissances, ce formalisme devant être aussi générique que possible, mais sémantiquement clair.

Le modèle obtenu est souvent qualifié de semi-formel (car certaines connaissances ne peuvent pas être totalement formalisées). Le caractère semi-formel d'une ontologie lui interdit d'être utilisée telle quelle dans un SBC. En revanche, une ontologie, contenant toutes les connaissances d'un domaine, constitue le support idéal de communication et de partage des connaissances de ce domaine.

4.4 Opérationnalisation

Cette étape consiste à formaliser complètement l'ontologie obtenue dans un langage de représentation de connaissances formel (i.e. possédant une syntaxe et une sémantique) et opérationnel (i.e. doté de services inférentiels permettant de mettre en œuvre des raisonnements), par exemple, le modèle des Graphes Conceptuels ou la Logique de Descriptions.

On obtient alors une représentation formelle des connaissances du domaine. Ainsi, le caractère formel de l'ontologie permet à une machine, via cette ontologie, de manipuler des connaissances du domaine. La machine doit donc pouvoir utiliser des mécanismes opérant sur les représentations de l'ontologie.

5 QUELQUES METHODOLOGIES DE CONSTRUCTION D'ONTOLOGIES

Les méthodologies peuvent porter sur l'ensemble du processus et guider l'ontologiste dans toutes les étapes de la construction. Bien qu'aucune méthodologie générale n'ait pour l'instant

réussi à s'imposer, de nombreux critères de construction d'ontologies ont été proposés pour des méthodologies.

ENTERPRISE, TOVE et METHONTOLOGY sont les méthodologies les plus représentatives pour construire des ontologies.

5.1 TOVE :

TOVE (Toronto Virtual Enterprise) développé par l'université de Toronto, cette méthodologie repose sur les expériences de développement d'une entreprise **[Gruber95, Uschold96]**. Elle s'appuie également, pour le développement d'une ontologie, sur les principales étapes suivantes :

- **Capturer des scénarios de motivations :** Cette étape consiste à identifier des scénarios qui clarifient le domaine que l'on investit et les différentes applications dans lesquelles l'ontologie sera employée.
- **Formuler des questions de compétences informelles :** Cette étape consiste à formuler un ensemble de questions (basées sur les scénarios), exprimées en langage naturel, afin de déterminer la portée de l'ontologie. Ces questions et leurs réponses sont utilisées pour extraire les concepts principaux, leurs propriétés et les relations qui existent entre ces concepts.
- **Spécifier la terminologie de l'ontologie :** Cette étape consiste à représenter les termes (concepts, propriétés et relations), identifier dans l'étape précédente, en utilisant le formalisme de la logique du premier ordre. Les concepts seront représentés sous forme de constantes ou bien des variables. Par ailleurs, les propriétés et les relations seront représentées par des prédicats.
- **Evaluer la complétude de l'ontologie.**

5.2 ENTERPRISE :

Uschold **[Uschold96]**, propose le squelette d'une méthode basé sur l'expérience de construction d'ontologies dans le domaine de la gestion des entreprises. La méthode ENTERPRISE repose sur les quatre étapes suivantes :

- Identifier le rôle et la portée de l'ontologie.
- Identifier les concepts et relations fondamentaux et des définitions provisoires de ces éléments. Coder l'ontologie dans un langage adapté. Intégrer des ontologies existantes. Dans cette étape, l'ontologie est réellement construite.
- Evaluer l'ontologie.
- Rédiger une documentation et une trace des actions réalisées lors des différentes phases .

Les étapes et sous-tâches de la méthode ENTERPRISE, sont décrites de façon abstraite. Les techniques utilisées pour les sous-tâches ne sont pas précisées (par exemple : Comment identifier les concepts fondamentaux ? Quel langage utiliser pour représenter l'ontologie ?

5.3 METHONTOLOGY :

La méthodologie de construction d'ontologies « METHONTOLOGY » se situe entre le GL (Génie Logiciel) et l'IC (Ingénierie des Connaissances). Elle identifie une séquence d'activités techniques à appliquer pour le développement de l'ontologie. L'approche METHONTOLOGY distingue les étapes suivantes :

5.3.1 Spécification :

Le développement d'une ontologie commence par la définition du domaine et portée de celle-ci. Cela est basé sur la réponse à certaines questions : Quel est le domaine que l'ontologie va couvrir ? À quoi cette ontologie va servir ? À quels types de questions les informations de l'ontologie doivent fournir des réponses ? Qui va utiliser et maintenir l'ontologie ?, etc. Les réponses à ces questions peuvent changer durant le processus de développement de l'ontologie, mais à chaque étape, elles permettent de limiter la portée du modèle. L'une des solutions qui permet de déterminer la portée d'une ontologie consiste à définir ou planifier une liste de questions auxquelles une base de connaissance, basée sur l'ontologie, doit être capable de répondre (*competency questions*) **[Gruber95]**.

5.3.2 Conceptualisation :

Elle consiste à identifier et à structurer les connaissances du domaine, à partir des sources d'informations. L'acquisition de ces connaissances peut s'appuyer à la fois sur l'analyse de documents et sur l'interview des experts du domaine. Une fois que les concepts sont identifiés par leurs termes, leur sémantique est décrite dans un langage semi-formel (tables et graphes) à travers leurs propriétés, leurs instances connues et les relations qui les lient entre eux.

5.3.3 Implémentation :

Cette étape consiste à formaliser le modèle conceptuel obtenu dans l'étape précédente par un formalisme de représentation d'ontologie telles que les logiques de description. Puis, à coder l'ontologie dans un langage d'ontologie formel.

5.3.4 Maintenance :

Cela peut s'agir d'une maintenance corrective ou évolutive de l'ontologie (nouveaux besoins de l'utilisateur), ce qui permet la validation et l'évolution de celle-ci. Cette activité est généralement faite par le constructeur et des experts du domaine. La validation se base sur l'exploitation des services d'inférences associés aux LDs, et qui sont offerts par des raisonneurs.

6 FORMALISMES DE REPRÉSENTATION

Représenter des connaissances propres à un domaine particulier consiste à décrire et à coder les entités de ce domaine de manière à ce qu'une machine puisse les manipuler afin de raisonner. Comme alternative à la logique classique, l'IA a proposé divers formalismes de représentation : ceux qui ont été le plus utilisés pour représenter les ontologies sont :

- Les frames
- Les graphes conceptuels

- Les logiques de description

6.1 Frames :

Le formalisme frames est introduit par M. Minsky **[Minsky75]**. Dans ce formalisme, la structure de données *enregistrement* représente une situation et un objet. L'idée est de collecter toutes les informations nécessaires concernant une situation et de les mettre dans une place, appelée frame. Certains auteurs, par exemple Hayes **[Hayes79]**, ont critiqué l'absence d'une sémantique formelle dans ce formalisme et ont montré qu'une fois convenablement formalisé, ce formalisme est une variante syntaxique de la logique de prédicat du premier ordre L.P.P.O.

Le modèle des Frames a été initialement proposé comme langage de représentation d'ontologies par T. GRUBER. Le principe de ce modèle est de décomposer les connaissances en classes (frames) qui représentent les concepts du domaine. À un frame est rattaché un certain nombre d'attributs (slots), chaque attribut peut prendre ses valeurs parmi un ensemble de facettes (facets). Une autre façon de présenter ces attributs est de les considérer comme des relations binaires entre classes dont le premier argument est appelé domaine (domain) et le deuxième est appelé portée (range) **[Gruber95]**.

6.2 Graphes conceptuels :

Le modèle des Graphes Conceptuels (GC), introduit par J. SOWA **[Sowa84]** au début des années 80, est un modèle opérationnel de représentation de connaissances, qui appartient à la famille des réseaux sémantiques.

Le formalisme réseaux sémantique est développé par M. Quillian pour représenter la sémantique du langage naturel **[Quillian68]**. Ce formalisme a une représentation de graphe dont les nœuds représentent des concepts et des individus. Ces nœuds sont connectes par des arcs étiquetés. Il y a deux sortes d'arcs : les arcs de propriété qui affectent les propriétés à des concepts ou a des individus et les arcs IS-A qui introduisent les relations hiérarchiques entre des concepts ou entre des individus.

Le modèle des GCs est mathématiquement fondé sur la logique et la théorie des graphes **[Sowa84]**. Cependant, pour raisonner à l'aide de GC, deux approches peuvent être distinguées : (1) raisonner à l'aide de la logique en considérant les GCs comme une interface graphique et (2) considérer les GCs comme un modèle de représentation à part entière disposant de ses propres mécanismes de raisonnement fondés sur la théorie des graphes.

Le modèle des GCs se décompose en deux parties **[Sowa84]** :

Une partie terminologique dédiée au vocabulaire conceptuel des connaissances à représenter, c'est-à-dire les types de concepts, les types de relations et les instances des types de concepts. Cette partie correspond à la représentation du modèle conceptuel mais intègre également des connaissances sur la hiérarchisation des types de concepts et de relations.

Une partie assertionnelle dédiée à la représentation des assertions du domaine de connaissance étudié.

6.3 Logique de description LD :

L'objectif principal des LDs consiste à pouvoir raisonner efficacement pour minimiser les temps de réponse. Par conséquent, la communauté scientifique a publié de nombreuses recherches qui portent sur l'étude du rapport expressivité/performance des différentes LDs **[Nardi03]**.

6.3.1 Historique :

Le développement des LDs fut fortement influencé par les travaux sur la logique des prédicats, les schémas (frames) **[Minsky81]** et les réseaux sémantiques. Des correspondances existent entre les LDs et ces formalismes **[Sattler03, Baader03a]**.

Les premiers travaux sur les LD commencèrent au début des années 1980 avec des systèmes à base de connaissances tels que KL-ONE, BACK et LOOM **[Baader03b, Nardi03]**.

Dans les années 1990, une nouvelle classe d'algorithmes est apparue : les algorithmes de vérification de satisfiabilité à base de tableaux. Ces derniers raisonnent sur des LD dites expressives ou très expressives, mais en temps exponentiel. Cependant, en pratique, le comportement des algorithmes est souvent acceptable **[Baader03b]**. L'expressivité accrue a ouvert la porte à de nouvelles applications telles que le Web sémantique **[Baader03b; Zou04; Horrocks03].** Le terme logiques de description expressives (LDE) désigne l'ensemble des LDs qui ont émergé pendant cette période.

6.3.2 Les deux niveaux de description :

La modélisation des connaissances d'un domaine avec les LDs se réalise en deux niveaux. Le premier, le niveau terminologique ou TBox, décrit les connaissances générales d'un domaine alors que le second, le niveau assertionnel ou ABox, représente une instanciation spécifique. Une TBox comprend la définition des concepts et des rôles, alors qu'une ABox décrit les individus en les nommant et en spécifiant en terme de concepts et de rôles, des assertions qui portent sur ces individus nommés.

TBox	ABox
Femelle ⊑ ⊤ ⊓ ¬Mâle	Humain(Anne)
Mâle ⊑ ⊤ ⊓ ¬Femelle	Femelle(Anne)
Animal ≡ Mâle ⊔ Femelle	Femme(Sophie)
Humain ⊑ Animal	Humain(Robert)
Femme ≡ Humain ⊓ Femelle	¬Femelle(Robert)
Homme ≡ Humain ⊓ ¬Femelle	Homme(David)
Mère ≡ Femme ⊓ ∃relationParentEnfant	relationParentEnfant(Sophie, Anne)
Père ≡ Homme ⊓ ∃relationParentEnfant	relationParentEnfant(Robert, David)
MèreSansFille ≡ Mère ⊓ ∀relationParentEnfant. ¬Femme	
relationParentEnfant ⊑ $\top_R$	

Tableau 2 – Une base de connaissances composée d'une TBox et d'une ABox

6.3.2.1 Le niveau terminologique (TBox)

Les concepts atomiques et rôles atomiques constituent les entités élémentaires d'une TBox. Les noms débutant par une lettre majuscule désignent les concepts, alors que ceux débutant par une lettre minuscule dénomment les rôles (par exemple : les concepts *Femelle*, *Mâle*, *Homme* et *Femme*, et le rôle *relationParentEnfant*).

6.3.2.2 Le niveau assertionnel (ABox)

Une ABox contient un ensemble d'assertions sur les individus. Chaque ABox doit être associée à une TBox, car les assertions s'expriment en terme de concepts et de rôles de la TBox. Une ABox désigne des individus caractérisés par des assertions d'individus nommés.

Une assertion de rôle, de la forme *R(a, b)* indique que pour cette ABox qu'il existe un individu nommé *a* qui est en relation avec un individu nommé *b* par le rôle *R* (défini dans la TBox associée).

6.3.3 La logique AL :

Cette logique, qui a été introduite par **[Schmidt-Schauss91]**, a étendu la logique FL^- **[Brachman84]** en y ajoutant la notion la négation des concepts atomiques. AL est minimale, dans le sens où une logique moins expressive représente peu d'intérêt **[Baader03a]**. Le Tableau 3 illustre les constructeurs offerts par AL.

Constructeur		description
C, D →	$\top$ |	Concept universel
	$\bot$ |	Concept le plus spécifique
	A |	Concept atomique
	$\neg$ A |	Négation atomique
	C $\sqcap$ D	intersection
	$\exists$r. $\top$	restriction existentielle limitée
	$\forall$r.C	restriction universelle

Tableau 3 – Les constructeurs selon AL

6.3.4 Les extensions d'AL :

Les logiques de description qui existent sont des combinaisons des différents éléments du Tableau 4. Par exemple, si on rajoute la négation complète C à la logique AI, on obtient la logique ALC.

Lettre	Constructeur	Syntaxe	Sémantique
$\mathcal{AL}$	nom de concept	C	$C^{\mathcal{I}}$
$\mathcal{AL}$	top	$\top$	$\Delta^{\mathcal{I}}$
$\mathcal{C}$	négation de concept non nécessairement primitif	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\mathcal{AL}$	conjonction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
$\mathcal{U}$	disjonction	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
$\mathcal{AL}$	quantificateur universel	$\forall R.C$	$\{d_1 \in \Delta^{\mathcal{I}} \mid \forall d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}}(d_1, d_2) \rightarrow d_2 \in C^{\mathcal{I}})\}$
$\mathcal{E}$	quantificateur existentiel typé	$\exists R.C$	$\{d_1 \in \Delta^{\mathcal{I}} \mid \exists d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}}(d_1, d_2) \wedge d_2 \in C^{\mathcal{I}})\}$
$\mathcal{N}$	restriction de nombre	$(\geq n\ R)$ $(\leq n\ R)$	$\{d_1 \in \Delta^{\mathcal{I}} \mid \{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\| \geq n\}$ $\{d_1 \in \Delta^{\mathcal{I}} \mid \{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\| \leq n\}$
$\mathcal{Q}$	restriction de nombre qualifiée	$(\geq n\ R.C)$ $(\leq n\ R.C)$	$\{d_1 \in \Delta^{\mathcal{I}} \mid \{d_2 \mid R^{\mathcal{I}}(d_1, d_2), d_2 \in C^{\mathcal{I}}\}\| \geq n\}$ $\{d_1 \in \Delta^{\mathcal{I}} \mid \{d_2 \mid R^{\mathcal{I}}(d_1, d_2), d_2 \in C^{\mathcal{I}}\}\| \leq n\}$
$\mathcal{O}$	un-de	$\{a_1, \ldots, a_n\}$	$\{d \in \Delta^{\mathcal{I}} \mid d = a_i^{\mathcal{I}}$ pour un $a_i\}$
$\mathcal{B}$	rôle filler	$\exists R.\{a\}$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d, a^{\mathcal{I}})\}$
$\mathcal{AL}$	nom de rôle	R	$R^{\mathcal{I}}$
$\mathcal{R}$	conjonction de rôles	$R_1 \sqcap R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$
$\mathcal{I}$	rôles inverses	R^{-1}	$\{(d_1, d_2) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d_2, d_1)\}$
$\mathcal{H}$	hiérarchie des rôles	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
$\mathcal{R}^+$	transitivité des rôles	R^+	Plus petite relation transitive contenant $R^{\mathcal{I}}$

Tableau 4 – Les constructeurs des LDs

Certaines logiques sont équivalentes, notamment ALC et $ALU\mathcal{E}$. Ces deux logiques augmentées par R^+ sont notées S. Les langages utilisés par OWL en sont une extension, respectivement $SHIF$ pour OWL Lite et $SHOIN$ pour OWL DL.

6.3.5 L'inférence :

L'inférence s'effectue au niveau terminologique ou assertionnel (factuel) :

- L'inférence au niveau terminologique comprend quatre principaux problèmes **[Baader03a]** : la satisfiabilité, la Subsomption, l'équivalence, et la disjonction.
- L'inférence au niveau assertionnel comprend quatre principaux problèmes aussi **[Baader03a]** : la Cohérence, la vérification d'instance, La vérification de rôle, et le problème de récupération.

7 OUTILS DE DEVELOPPEMENT D'ONTOLOGIES

Les outils de développement d'ontologies qui existent sur le marché aujourd'hui sont divers et variés. Cet état de choses suscite beaucoup d'interrogations lorsque vient le moment d'en choisir un pour construire une nouvelle ontologie **[Gomez-Pérez02]** : L'outil offre-t-il une assistance au développement ? L'outil dispose-t-il d'un moteur d'inférence ? Quels langages d'ontologies l'outil supporte-t-il ? L'outil permet-il d'importer/exporter des ontologies ? L'outil offre t-il un support à la réutilisation d'ontologies existantes ? L'outil permet-il de documenter les ontologies construites ? L'outil offre-t-il un support graphique à la construction des ontologies ? L'outil est-il stable, convivial, « mature » ? Les réponses à toutes ces questions pourraient s'avérer décisives dans le choix de l'un ou l'autre outil. Dans cette section nous passons en revue les principaux outils disponibles.

7.1 Langages de spécification d'ontologies :

Dans le contexte du Web sémantique, plusieurs langages d'ontologies ont été développés pendant les dernières années. Certains d'entre eux sont basés sur la syntaxe de XML, tels que XOL, SHOE, OML, RDF, et RDF Schéma. Les deux derniers sont créés par des groupes de travail du W3C. Trois autres langages sont établis sur RDF(S) pour améliorer ses caractéristiques : OIL, DAML+OIL et OWL. La Figure 7 représente la pyramide des langages du Web sémantique.

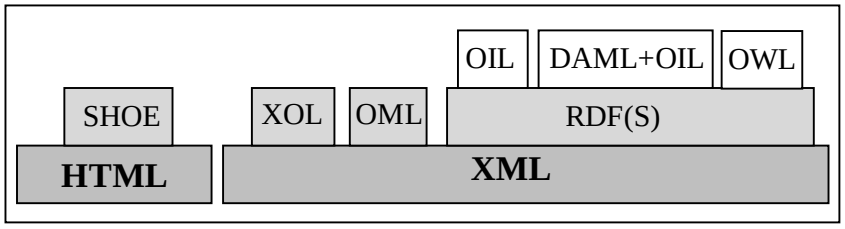

Figure 7 – La pyramide des langages du Web sémantique

7.1.1 RDF :

RDF **[W3C04a]** est un langage pour la représentation de méta-données à propos des ressources. Le modèle RDF permet cette représentation par des **assertions** sous la forme d'un triplet (*ressource*, *propriété*, *valeur*), ou encore (*sujet*, *prédicat*, *objet*) :

- **Ressources** : les ressources sont tous les objets *décrits* par RDF. Généralement, ces ressources peuvent être aussi bien des pages Web que tout objet ou personne du monde réel. Les ressources sont alors identifiées par leur URI (Uniform Resource Identifier).

- **Propriétés** : une propriété est un attribut, un aspect, une caractéristique qui s'applique à une ressource. Il peut également s'agir d'une mise en relation avec une autre ressource.
- **Valeurs** : les valeurs en question sont les valeurs particulières que prennent les propriétés. La valeur pouvant être une autre ressource ou bien un littéral.

Exemple : Sami a 23 ans et habite Constantine

```
<rdf :RDF>
<rdf :Description about='Sami'>
<rdf :Property about='ville'>
    Constantine
</rdf :Property>
<rdf :Property about='age'>
    23
</rdf :Property>
</rdf :Description>
</rdf :RDF>
```

7.1.2 RDF(S) :

RDFS **[W3C04b]** est un langage permettant de définir des schémas de méta-données. Il définit le sens, les caractéristiques et les relations d'un ensemble de propriétés. La principale nouvelle notion est la distinction entre une classe (concept d'une ontologie) et une instance (individu d'une ontologie). Quelques notions définies sont : (rdfs : Class), (rdfs : subClassOf), (rdfs : domain), et (rdfs : range).

Sur l'exemple de Sami, nous définissons le concept de personne, une taxinomie de concepts, et l'instance Sami.

```
<rdf :RDF>
<rdfs :Class rdf :about='Personne'>
<rdfs :subClassOf rdf :resource='Thing'/>
</rdfs :Class>
<rdf :Property about='age'>
<rdfs :domain rdf :resource='Personne'/>
<rdfs :range rdf :resource='xsd :integer'/>
</rdf :Property>
<rdf :Property about='ville'>
<rdfs :domain rdf :resource='Personne'/>
<rdfs :range rdf :resource='xsd :string'/>
</rdf :Property>
</rdf :RDF>
```

```
<Personne rdf :ID='Sami'>
<age rdf :resource='23' />
<ville rdf :resource='Constantine' />
</Personne>
```

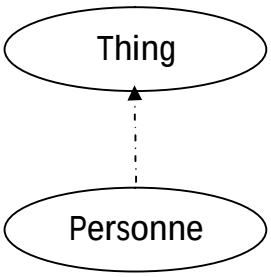

Figure 8 – Taxinomie de concept personne

7.1.3 DAML-OIL :

DAML[1] est un langage qui a comme but de fournir les fondations pour la génération suivante du Web sémantique. Comme RDFS, ce langage n'est pas assez expressif relativement aux exigences du Web sémantique, un nouveau langage nommé DAML-ONT a été développé en tant qu'extension de RDF avec les capacités d'un langage de représentation du savoir.

[1] The DARPA Agent Markup Language. http://www.daml.org/.

En même temps, nouveau langage nommé OIL a été développé par un groupe des chercheurs pour le même but. Ce langage a une syntaxe basée sur RDF et il est explicitement construit pour que sa sémantique puisse être spécifiée à travers une description logique très expressive, la logique de description de type SHIQ.

DAML+OIL[2] est la combinaison de ces deux langages. Il hérite des avantages de ces deux langages. En conséquence, DAML+OIL est un langage très expressif et lisible par la machine ainsi que par un être humain avec une syntaxe basée sur RDF.

7.1.4 OWL :

OWL **[W3C04c]** est un langage fondé sur la syntaxe RDF/XML et héritier des travaux de DAML+OIL. OWL introduit l'aspect sémantique qui manque RDF, et offre, par ses primitives plus riches, au machine une capacité d'interprétation plus grande que celle de RDF et RDFS.

OWL se compose de trois sous-langages OWL Lite, OWL DL et OWL Full, qui offrent des capacités d'expression croissantes, chacun est une extension par rapport à son prédécesseur plus simple.

OWL Lite est le sous langage le plus simple, il répond à des besoins de hiérarchie de classification et de fonctionnalités de contraintes simples. Le Tableau 5 ci-dessous présente les différents constructeurs de ce langage.

Catégorie	Constructeurs	Exemple
RDF Schema	Class, rdfs :subClassOf, rdf :Property, rdfs :subPropertyOf, rdfs :domain, rdfs :range, Individual	Personne : subClassOf(Thing) Femme : subClassOf(Personne) Enfant : subClasseOf(Personne)
In(Equalité)	equivalentClass, equivalentProperty, sameAs, differentFrom, AllDifferent, distinctMembers	Fille : equivalentClass(intersectionOf(Femme, Enfant))
Restrictions	onProperty, allValuesFrom, someValuesFrom	Parent : intersectionOf (Personne, restriction(minCardinality(1), onProperty(aEnfant))
Cardinalités (0 ou 1)	minCardinality, maxCardinality, cardinality	
Intersection	intersectionOf	
Propriétés	SymmetricProperty,FunctionalProperty, ObjectProperty, DatatypeProperty, inverseOf, TransitiveProperty, InverseFunctionalProperty	aParent : ObjectProperty (Enfant, Personne) aEnfant : inverseOf (aParent)

Tableau 5 – Les constructeurs de OWL Lite

OWL DL est plus complexe que OWL Lite, il est fondé sur la logique de description $SHIN(D)$. WL DL garantit la complétude des raisonnements (calculabilité des inférences) et leur décidabilité (leur calcul se fait en une durée finie). Le Tableau 6 montre la liste des constructeurs ajoutés par OWL DL par rapport à OWL Lite.

[2] DAML est utilisé pour l'aspect description de connaissances et OIL pour le côté raisonnement, parmi les auteurs de ces travaux on peut citer Ian Horrocks et Tim Berners-Lee.

Catégorie	Constructeurs	Exemple
Axiome de Classe	oneOf (enumération), dataRange, disjointWith	Gender : oneOf(Male, Female)
Expressions booléennes	unionOf, complementOf	Tante : intersectionOf (Femme, unionOf (aNeuve, aNiece))
Cardinalité (0, n)	minCardinality, maxCardinality, cardinality	
Individu cible d'une propriété	hasValue	Homme : intersectionOf (Personne, hasValue(sexe,Male))

Tableau 6 – Liste des constructeurs ajoutés par OWL DL

OWL Full est la version la plus complexe d'OWL, il se destine aux utilisateurs souhaitant une expressivité maximale. Il a l'avantage de la compatibilité complète avec RDF/RDFS, mais l'inconvénient est qu'il ne garantit pas la complétude et la décidabilité des calculs liés à l'ontologie.

7.2 Les moteurs d'inférences :

La plupart de ces moteurs acceptent en entrée des fichiers OWL et sont conçus pour raisonner sur les logiques de descriptions. Une fois l'ontologie chargée, ces moteurs effectuent les inférences sur la TBox et la ABox. Le tableau ci-dessous dresse une comparaison des principaux moteurs d'inférence pour les logiques de description expressives : FaCT**[Horrocks98]**, Racer**[Haarslev01]**, Pellet**[Sirin06]**, FaCT++**[Tsarkov04]**, F-OWL, Surnia, et Hoolet.

Moteur	Logique de Description	Implantation	Inférence	API Java	OWL	Décidabilité
Racer	$SHIQ$	C++	TBox/ABox	oui	OWL DL	oui
Pellet	$SHIN(D)$, $SHON$	Java	TBox/ABox	natif	OWL DL	oui
FaCT	$SHIQ$, SHF	Common Lisp	TBox	oui	OWL DL	oui
FaCT++	$SHIF$	C++	TBox	oui	OWL-Lite	oui
Surnia	Logique prédicats	Python	TBox/ABox	non	OWL Full	non
Hoolet	Logique prédicats	Java	TBox/ABox	oui	OWL DL	non
F-OWL	$SHIQ$	Java	TBox/ABox	oui	OWL Full	non

Tableau 7 – Les principaux moteurs d'inférence

7.2.1 Racer :

Racer est le moteur d'inférence le plus connu et l'un des plus utilisés dans le domaine pour ces performances et sa stabilité. Il est commercialisé par Racer Systems GmbH & Co. KG, fondé en 2004 par Volker Haarslev, Kay Hidde, Ralf Möller et Michael Wessel qui travaillaient à l'université de Hambourg. Racer travaille sur les ontologies modélisées par son langage, mais il

accepte des ontologies décrites en RDF ou OWL, ces dernières étant traduites vers le langage utilisé par Racer. Ce moteur d'inférence possède également son propre langage de requête nRQL (new Racerpro query Language) pour interroger l'ontologie sur la TBox et ABox.

Racer possède quelques avantages :

- Racer permet l'ajout d'assertions et d'individus dans les ABox après le chargement de l'ontologie.
- Racer permet l'utilisation de règles SWRL.

Racer possède quelques points négatifs :

- Racer suppose que tous les propriétés sur les datatypes sont fonctionnelles (pas de valeurs multiples pour un datatype property).
- Racer ne permet pas l'utilisation de type de données utilisateur (type défini par l'utilisateur), et il n'existe pas de version libre d'utilisation. Cependant il est possible d'obtenir une licence gratuite dans le cadre de la recherche scientifique.

7.2.2 Pellet :

Le moteur Pellet **[Sirin06]** est beaucoup plus récent. Pellet est un des projets du MINDSWAP Group, un groupe de recherche sur le web sémantique de l'université du Maryland.

Il est disponible en OpenSource et offre des évolutions fréquentes. Pellet travaille sur des ontologies décrites en RDF ou OWL et permet les requêtes avec RDQL et SPARQL sur la ABox et la TBox.

Pellet possède quelques avantages :

- Pellet est open-source et développé en Java.
- Pellet est un raisonneur OWL DL complet.
- Pellet propose en cas d'incohérence dans l'ontologie des réparations possibles.

Pellet possède quelques points négatifs :

- Pellet possède une documentation pauvre en comparaison de celle de Racer. En effet racer est le plus utilisé et donc le plus documenté par des particuliers.
- Pellet n'offre pas de système de souscription à un concept.

7.3 Langages d'interrogation d'ontologies :

Dans cette section, nous présentons les trois langages d'interrogation basés sur la reconnaissance de graphe RDF : RDQL, SPARQL et nRQL. Le principe est de décrire un graphe RDF à l'aide de variables. Les résultats d'une requête étant les valeurs des variables pour lesquelles il existe un graphe RDF dans l'ontologie correspondant au graphe défini par la requête. RDQL et SPARQL sont les langages d'interrogation utilisables sur Pellet et nRQL est le langage d'interrogation de Racer.

7.3.1 RDQL :

RDQL (*RDF Data Query Language*) **[W3C04d]** est un langage d'interrogation de données définies en RDF. Ce langage n'est pas standardisé, il existe de nombreuses implémentations. Sa syntaxe est très proche de SQL (*Structured Query Language*) :

```
SELECT variable [, variable]*
FROM documents_rdf [, documents_rdf]*
WHERE modele_de_triplets
AND restrictions_booléennes
USING définition_des_raccourcis
```

- La clause SELECT définit la liste des variables que l'on désire obtenir. Une variable est composée de caractères alphanumériques et commence par un ' ?'.
- La clause FROM définit l'emplacement des documents RDF utilisés pour la requête.
- La clause WHERE définit le triplet RDF (sujet - prédicat - objet), les éléments de ce triplet sont décrits soit par les valeurs de l'ontologie interrogée soit par des variables.
- La clause AND définit les restrictions booléennes de la requête. Une restriction booléenne est constituée de valeurs ou variables composé à l'aide d'opérateurs :

Voici un exemple de requête RDQL, qui sélectionne l'âge d'une personne dans le document annuaire.rdf sachant que la personne possède un âge inférieur ou égal à 50 ans, que son nom est Dujardin et son adresse email est **Dujardit@lifl.fr**.

```
SELECT ?name, ?email, ?age
FROM <http ://www.lifl.fr/ANNUAIRE/annuaire.rdf>
WHERE ( ?email, vcard :EMAIL, "Dujardit@lifl.fr")
AND ( ?age <= 50 ) && ( ?name EQ "Dujardin") )
WHERE ( ?email, vcard :EMAIL, "Dujardit@lifl.fr")
USING vcard FOR <http ://www.w3.org/2001/vcard-rdf/3.0/>
```

7.3.2 SPARQL :

SPARQL **[W3C06]** est une amélioration de RDQL, ce langage est en cours de standardisation au niveau du W3C. SPARQL offre une syntaxe quasi identique à RDQL, mais y ajoute notamment les opérateurs UNION et OPTIONAL dans la clause WHERE. L'opérateur UNION définit la disjonction de triplets RDF.

7.3.3 nRQL :

nRQL (new Racerpro query Language) **[Haarslev01]** est le langage d'interrogation de Racer. Comme RDQL et SPARQL, nRQL est basé sur la recherche de graphes RDF. Sa syntaxe est proche des deux autres, sauf pour sa notation préfixée des opérateurs. Ci-dessous, nous présentons une requête en nRQL cherchant tous les oncles (variable z) dans l'ontologie myOntology.

```
RETRIEVE ($ ?z)
(AND ( $ ?x $ ?y |myOntology#aEnfant|)
($ ?z $ ?x |myOntology#estFrereDe|))
```

7.4 Editeur d'ontologies :

Aujourd'hui, le nombre d'éditeurs d'ontologies a considérablement augmenté. Il existe différents éditeurs qui utilisent des formalismes variés et offrent différentes fonctionnalités. Parmi ces outils on trouve : OILed, OntoEdit, , Web ode, DOE, Protégé2000… voici la description de quelques-uns :

7.4.1 OILEd

OILEd **[Bechhofer01]** a été conçu pour éditer des ontologies dans le langage de représentation OIL, il est souvent considéré comme une simple interface de la logique de description SHIQ. Cet éditeur offre également les services d'un raisonneur, FaCT qui permet de tester la satisfiabilité des définitions de classes et de découvrir des subsomptions restées implicites dans l'ontologie. L'outil dispose de mécanismes pour la classification et le contrôle de la cohérence des ontologies. La version 3.4 d'OILEd est gratuite et disponible sur le site web d'OILEd.

7.4.2 OntoEdit

OntoEdit **[Sure02]** est également un environnement de construction d'ontologies indépendant de tout formalisme. Des outils graphiques dédiés à la visualisation d'ontologies sont inclus dans l'environnement. ONTOEDIT intègre un serveur destiné à l'édition d'une ontologie par plusieurs utilisateurs. Un contrôle de la cohérence de l'ontologie est assuré à travers la gestion des ordres d'édition.

7.4.3 Ontolingua

Ontolingua **[Farquhar96]** de l'Université Stanford. Le serveur Ontolingua est le plus connu des environnements de construction d'ontologies en langage Ontolingua. Il consiste en un ensemble d'environnements et de services qui supportent la construction en coopération d'ontologies, entre des groupes séparés géographiquement. Il supporte plusieurs langages et dispose de traducteurs permettant de passer de l'un à l'autre.

7.4.4 ONTOSAURUS

ONTOSAURUS **[Swartout97]** de l'Information Science Institute de l'Université de Southern California. Ontosaurus consiste en un serveur utilisant LOOM comme langage de représentation des connaissances, et en un serveur de navigation créant dynamiquement des pages HTML qui affichent la hiérarchie de l'ontologie; le serveur utilise des formulaires HTML pour permettre à l'usager d'éditer l'ontologie. Des traducteurs du LOOM en Ontolingua, KIF, KRSS et C++, ont été développés.

7.4.5 Protégé2000

Protégé2000 **[Noy01]** est une interface modulaire, développée au Stanford Medical Informatics de l'Université de Stanford7, permettant l'édition, la visualisation, le contrôle (vérification des contraintes) d'ontologies, l'extraction d'ontologies à partir de sources textuelles, et la fusion semi-automatique d'ontologies. Le modèle de connaissances de Protégé2000 est issu du modèle des frames et contient des classes (concepts), des slots (propriétés) et des facettes (valeurs des propriétés et contraintes), ainsi que des instances des classes et des propriétés. De nombreux plug-ins sont disponibles ou peuvent être ajoutés par l'utilisateur.

8 LES ONTOLOGIES DANS LA SECURITE INFORMATIQUE

Dans ces dernières années, les ontologies dans la sécurité informatique sont largement utilisées vue le besoin d'un partage sémantique de connaissance. Nous allons présenter trois approches principales, qui sont les plus citées :

- RBAC (*Role-Based Access Control*)
- KAoS (*Knowledge Acquisition in autOmated Specification*)
- Rei (prononcé en anglais « ray », un mot japonais qui signifie "universel")

8.1 Contrôle d'accès à base de rôles (RBAC) :

RBAC est un modèle de contrôle d'accès à un système d'information dans lequel chaque décision d'accès est basée sur le rôle auquel l'utilisateur est attaché. Un rôle découle généralement de la structure d'une entreprise. Les utilisateurs exerçant des fonctions similaires peuvent être regroupés sous le même rôle.

La modification des contrôles d'accès n'est pas nécessaire chaque fois qu'une personne rejoint ou quitte une organisation. Par cette caractéristique, RBAC est considéré comme un système « idéal » pour les entreprises dont la fréquence de changement du personnel est élevée.

Les approches présentées dans cette section sont décrites en détail dans **[Sandhu98]**. L'idée centrale des approches de RBAC est que les permissions sont associées aux rôles, et les utilisateurs sont affectés aux rôles appropriés. Les rôles sont créés pour diverses fonctions dans une organisation et sont assignés aux utilisateurs selon leurs responsabilités et qualifications. On peut accorder de nouvelles permissions aux rôles à chaque fois que de nouvelles applications et nouveaux systèmes sont incorporés, et des permissions peuvent être retirées des rôles quand c'est nécessaire. La première série de ces approches (RBAC0) est basée sur de simples classes d'entités, de relations et de fonctions.

8.1.1 Classes d'entités :

Il y a quatre classes principales d'entités :

- **Les utilisateurs** : En général, les utilisateurs peuvent être considérés comme des agents : êtres humains, robots, agents logiciels, etc.
- **Les rôles :** Un rôle est une fonction au sein de l'organisation avec une certaine sémantique associée concernant l'autorité et la responsabilité d'un membre du rôle.
- **La permission :** Une permission est un accord d'un "mode d'accès" particulier à un ou plusieurs "objets" dans le système. Les permissions sont toujours positives et confient la capacité au détenteur d'accomplir quelques actions dans le système. RBAC0 considère les permissions en tant que symboles non-interprétés parce que la nature précise d'une permission dépend du système.
- **Les sessions :** chaque session est une correspondance d'un utilisateur à plusieurs rôles possibles, c.-à-d., un utilisateur établit une session pendant laquelle il active un certain sous-ensemble de rôles dont il est membre.

8.1.2 Relations RBAC0 :

Il y a deux relations principales :

- *UserAssignement* : Une relation entre l'utilisateur et le rôle de type plusieurs à plusieurs (un utilisateur peut avoir plusieurs rôles et le même rôle peut être affecté à plusieurs utilisateurs).
- *PermissionAssignment :* Une relation entre le rôle et la permission (privilège) de type plusieurs à plusieurs (la même permission peut être assignée à différents rôles et le même rôle peut avoir différentes permissions).

Il est possible que deux rôles puissent avoir exactement les mêmes permissions mais soient toujours des rôles séparés. Il en est de même pour les utilisateurs.

8.1.3 Fonctions RBAC0 :

Il y a deux fonctions principales :

- *user* : $S \rightarrow U$: Une fonction qui détermine pour chaque session s_i un seul utilisateur $user(s_i)$.
- *roles* : $S \rightarrow 2^R$: Une fonction qui détermine pour chaque session s_i un ensemble de rôles inclus dans l'ensemble de rôles attachés à l'utilisateur $user(s_i)$, c.-à-d. $roles(s_i) \subseteq \{r \mid (user(s_i), r) \in UA\}$.

8.1.4 RBAC1 et RBAC2 :

Les approches RBAC1 ajoutent l'hiérarchie de rôle. Une relation d'ordre partiel entre les rôles est ajoutée à RBAC0. Toutes les permissions du rôle général sont héritées par les rôles spécifiques.

Les approches RBAC2 exigent une collection de contraintes qui déterminent si des valeurs de divers composants de RBAC0 sont acceptables. Seules les valeurs acceptables seront autorisées. Un exemple classique de contrainte est l'exclusivité mutuelle entre les rôles : le même individu ne sera pas autorisé à être un membre des deux rôles. Les contraintes sont un mécanisme puissant pour imposer la politique d'organisation de plus haut niveau.

8.2 KAoS :

KAoS **[Uszok03]** est une approche de gestion d'agent basée sur la politique, comprenant les problèmes typiques de sécurité (les politiques d'autorisation, le cryptage, politique d'accès et de contrôle de ressources) et des notions supplémentaires (politiques de conversation d'agent, représentations et mécanismes d'application pour des politiques de mobilité, des politiques d'enregistrement du domaine, des politiques d'engagements, etc.).

8.2.1 Politique de KAoS :

Dans le domaine de gestion du réseau, Les politiques sont exprimées comme ensembles de règles qui décrivent le comportement du réseau. Dans KAoS, une politique est un rapport permettant ou contraignant l'exécution d'un certain type d'action par un ou plusieurs acteurs par rapport à divers aspects d'une certaine situation. Le système peut détecter et harmoniser

des conflits entre les politiques. Dans KAoS la distinction entre les autorisations (contraintes qui permettent ou interdisent une certaine action) et les engagements (contraintes qui exigent une certaine action) est clairement énoncée.

8.2.2 Ontologie de politique :

KAoS utilise des concepts d'ontologie encodés en OWL pour établir des politiques (voir http://ontology.ihmc.us/ontology). Ces politiques contraignent des actions permises exécutées par des acteurs (des clients ou des agents). Le service de politique de KAoS distingue les autorisations (c.-à-d., les contraintes qui permettent ou interdisent une certaine action) et les engagements (c.-à-d., les contraintes qui exigent d'une certaine action d'être exécutée). L'applicabilité de la politique est définie par une classe de situations contenant les composants spécifiant l'historique, l'état et l'action actuelle.

Les ontologies de politique de KAoS définissent des ontologies pour les actions, les acteurs, les groupes, les endroits, les diverses entités liées aux actions (par exemple, ressources informatiques), et des politiques. Il existe 79 classes et 41 propriétés définies en ontologies de base.

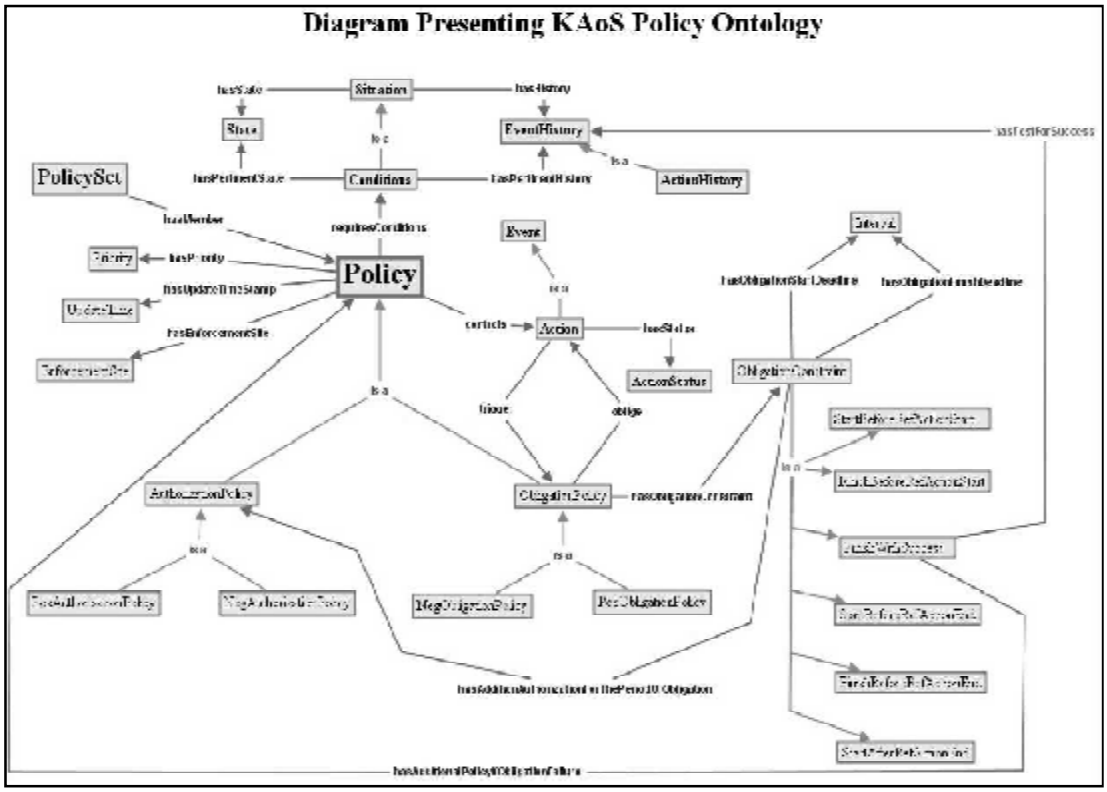

Figure 9 – Diagramme représentant l'ontologie KAoS [Uszok03]

- **Acteurs et groupes d'acteurs** : L'ontologie d'acteur distingue les personnes et les diverses classes des agents logiciels qui peuvent être les sujets de la politique.
- **Actions** : L'ontologie actions définit la relation entre les classes d'action telle que la relation *PartOf* (pour les actions composées) et la relation *implementedBy* entre les classes d'action abstraites et les exécutions dans une application. Les agents peuvent exécuter des actions ordinaires et/ou peuvent être autorisés ou obligés à exécuter certaines actions de politique.

8.3 Rei :

L'ontologie de politique de Rei est présentée en détails dans **[Kagal02].**

Rei se compose de plusieurs ontologies : ReiPolicy, ReiMetaPolicy, ReiEntity, ReiDeontic, ReiConstraint, ReiAnalysis, et ReiAction. Chaque ontologie décrit les classes et les propriétés liées à ce domaine.

Domaine	Ontologie	Liste de classes
Politique	ReiPolicy	Classe : ReiRoot, Policy, Granting
Métapolitique	ReiMetaPolicy	Classe : MetaPolicy, ModalityPrecedence, Behaviour, MetaMetaPolicy, Priority Sous-classe de Priority : RulePriority, PolicyPriority
Entité	ReiEntity	Classe : Entity Sous-classe de Entity : Agent, Object, Variable
Déontique	ReiDeontic	Classe : DeonticObject Sous-classe de DeonticObject : Permission, Obligation, Prohibition, Dispensation
Action	ReiAction	Classe : Action Sous-classe de Action : DomainAction, SpeechAct Sous-classe de SpeechAct : Delegation, Revocation, Obligation, Dispensation
Contrainte	ReiConstraint	Classe : Constraint Sous-classe de Constraint : SimpleConstraint, BooleanConstraint Sous-classe de BooleanConstraint : And, Or, Not
Analyse	ReiAnalysis	Classe : Analysis Sous-classe de Analysis : WhatIf, UseCase Sous-classe de WhatIf : WhatIfProperty, WhatIfPolicyRule Sous-classe de UseCase : StatementUseCase, DeonticUseCase

Tableau 8 – Liste de domaines, ontologies et classes de Rei

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-
    schema#" xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:entity="http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiEntity.owl#"
    xmlns="http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiEntity.owl#">
<owl:Ontology rdf:about="http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiEntity.owl#">
  <owl:versionInfo>2.0</owl:versionInfo>
  <rdfs:comment>Defines Entity, Object, Agent, and Variable</rdfs:comment>
</owl:Ontology>
<!-- ENTITY CLASS -->
    <owl:Class rdf:ID="Entity" rdfs:label="Entity">
    <rdfs:comment>All entities in the system</rdfs:comment>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="affiliation" rdfs:label="affiliation">
    <rdfs:domain rdf:resource="http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiEntity.owl#Entity" />
    </owl:ObjectProperty>
<!-- SUBCLASSES OF ENTITY CLASS -->
<!-- AGENT CLASS -->
    <owl:Class rdf:ID="Agent" rdfs:label="Agent">
    <rdfs:comment>All software or human agents</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiEntity.owl#Entity" />
    </owl:Class>
<!-- OBJECT CLASS -->
    <owl:Class rdf:ID="Object" rdfs:label="Object">
    <rdfs:subClassOf rdf:resource="http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiEntity.owl#Entity" />
    </owl:Class>
    <owl:ObjectProperty rdf:ID="owner" rdfs:label="owner">
    <rdfs:domain rdf:resource="http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiEntity.owl#Entity" />
    <rdfs:range rdf:resource="http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiEntity.owl#Entity" />
    </owl:ObjectProperty>
<!-- VARIABLE CLASS -->
    <owl:Class rdf:ID="Variable" rdfs:label="Variable">
    <rdfs:comment>A rei variable</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.cs.umbc.edu/~lkagal1/rei/ontologies/ReiEntity.owl#Entity" />
    </owl:Class>
</rdf:RDF>
```

Figure 10 – Une partie de l'ontologie Rei

8.4 Dispositifs des différentes approches :

Quelques comparaisons intéressantes entre les approches principales de l'ontologie de la sécurité (en particulier entre KAoS et Rei) sont contenues dans **[Tonti03]**. Nous pouvons citer les propriétés suivantes :

- Les approches de RBAC ont été créées pour les environnements statiques
- les approches basées sur KAoS fournissent des détails au sujet de la politique.
- Rei est plus approprié aux environnements ouverts et dynamiques

9 CONCLUSION

Les ontologies apparaissent désormais comme une clé pour la manipulation automatique de l'information au niveau sémantique. Au fur et à mesure des recherches, des idées se dégagent autour du contenu des ontologies, des méthodes à utiliser pour les construire et des modèles et langages servant à leur représentation.

Au long de ce chapitre, nous avons essayé d'éclaircir la notion d'ontologie en présentant certaines définitions. Nous avons découvert les méthodologies les plus représentatives de leur construction et quelques domaines de leur utilisation. Nous avons montré après les principaux formalismes de représentation de connaissances à savoir les frames, les graphes conceptuels et les logiques de descriptions. Nous avons aussi présenté les outils nécessaires à leur développement à savoir les langages de représentation, les outils d'édition et d'interrogation, …etc. Et finalement, nous avons présenté quelques ontologies utilisées dans le domaine de sécurité.

Le chapitre suivant présente notre contribution au problème posé par ce mémoire, à savoir la construction d'une ontologie pour le domaine de la sécurité. Pour cela, nous allons présenter, de façon générale, comment utiliser et appliquer cette ontologie aux agents mobiles par des scénarios proposés, ensuite, nous allons suivre un processus de construction de l'ontologie qui est l'objectif de ce mémoire.

CHAPITRE 3

CONSTRUCTION DE L'ONTOLOGIE

1 INTRODUCTION

Ce chapitre présente notre contribution au problème posé par ce mémoire qui est la construction d'une ontologie dédiée aux agents mobiles, dont le domaine est la sécurité. Pour ce faire, nous nous somme basé sur la méthodologie METHONTOLOGY **[Fernandez97]** qui est le support de base pour la conceptualisation de l'ontologie à créer, à travers un ensemble de représentations intermédiaires semi-formelles. La logique de descriptions, est le formalisme adopté pour l'expression de l'ontologie semi-formelle. OWL, le langage de définition d'ontologies, est choisi afin de codifier l'ontologie en utilisant l'éditeur d'ontologies Protégé OWL. Finalement, le système d'inférences RACER (*Renamed Abox and Concept Expression Reasoner*), est utilisé afin de tester la consistance de l'ontologie tout au long du processus de développement.

2 L'ONTOLOGIE MASO

Notre contribution, essentiellement, est une ontologie de domaine de sécurité des agents mobiles MASO (*Mobile Agent Security Ontology*).

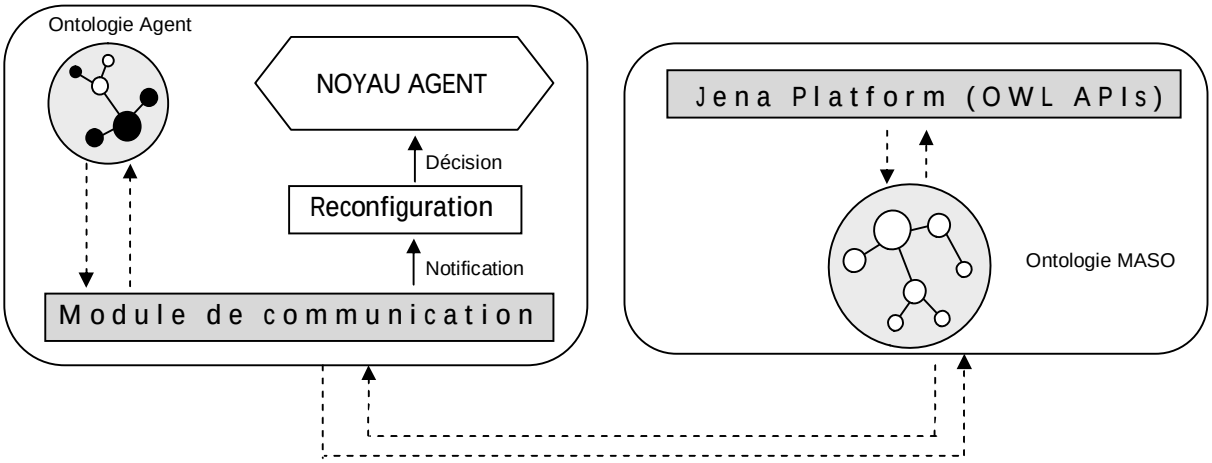

Figure 11 – Agent mobile sensible au contexte de sécurité

2.1 Contexte de sécurité :

Tout au long de sa migration, l'agent a besoin d'interagir avec les différents environnements visités. L'environnement, potentiellement hétérogène et imprévisible, peut influencer le processus d'exécution. Par conséquent, l'agent mobile doit être sensible à son contexte de sécurité afin de raisonner dessus et vérifier si les conditions requises et capacités acquises pour son exécution sont satisfaites.

La définition du contexte la plus complète et la plus adoptée est celle proposée dans **[Dey00]** :

Définition6 « *Le contexte désigne toute information qui peut être utilisée pour caractériser la situation d'une entité. Une entité peut être une personne, un endroit, ou un objet considérés comme pertinents pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application eux-mêmes* » **[Dey00]**.

[AmaraH06], dans le chapitre **5** section **1.2,** propose une définition du contexte d'un agent mobile GAMA :

Définition7 « *Le contexte d'un agent GAMA désigne l'ensemble des paramètres caractérisant son environnement d'exécution et qui peuvent influencer son comportement. Ces paramètres sont regroupés en trois familles représentant trois aspects différents du contexte, à savoir le contexte social, le contexte utilisateur et le contexte physique* » **[AmaraH06]**.

En s'inspirant des deux définitions précédentes, nous définissons le contexte de sécurité d'un agent mobile comme suit :

Définition8 « *Le contexte de sécurité d'un agent mobile désigne l'ensemble des paramètres de sécurité caractérisant son environnement d'exécution et qui peuvent influencer son comportement* »

2.2 Utilisation de l'ontologie :

Avec l'ontologie MASO, l'agent mobile peut raisonner sur le contexte de sécurité de l'entité en interaction. Parmi les différents paramètres de sécurité, nous citons :

- **Les exigences de sécurité :** l'agent et l'hôte peuvent raisonner sur les mesures de sécurité et les mécanismes utilisés. L'agent peut analyser le contexte de l'hôte et vice-versa pour décider si les exigences de l'agent ou de l'hôte sont vérifiées. (Une sous-ontologie concernant les objectifs et les mécanismes)
- **Les menaces de sécurité :** la sous-ontologie concernant les menaces et les risques est utilisée par l'agent mobile pour raisonner sur le contexte fourni par l'hôte ou un autre agent, et ainsi prendre une décision d'accepter ou refuser la communication ou l'interaction, ou d'accepter ou refuser d'être exécuté sur la plate-forme d'accueil.
- **Les problèmes de sécurité :** s'il y a un échec d'interopérabilité due à l'hétérogénéité du système ouvert (différents mécanismes, hétérogénéité structurelle des propriétés telle que les mots de passe, les algorithmes de cryptage, …) la sous-ontologie concernant les protocoles, les algorithmes supportés, … etc. peut donner une explication sur la raison de l'échec ? dans ce cas, l'agent mobile peut s'adapter, si c'est possible, aux différentes situations pour réussir l'interopérabilité.

2.2.1 Communication entre deux agents mobiles :

L'ontologie MASO, dans le cas où deux agents mobiles sont en communication, peut masquer l'hétérogénéité pouvant exister au niveau des propriétés de sécurité, et fournir une sémantique explicite pour que le contexte de sécurité soit compréhensible et interprétable.

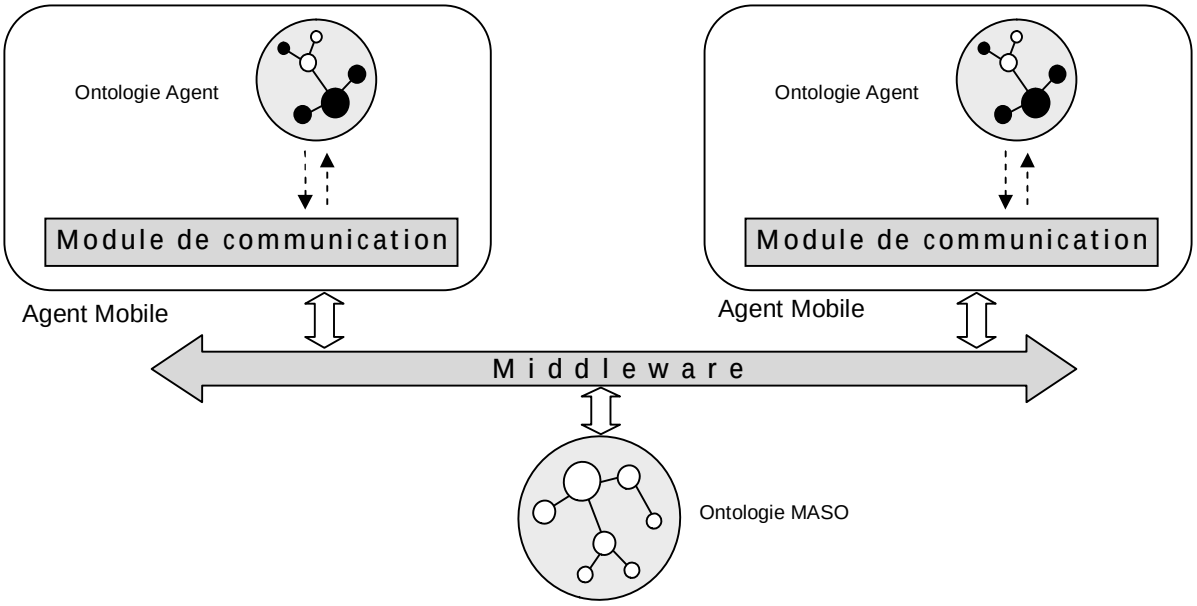

Figure 12 – Communication entre deux agents mobiles en utilisant l'ontologie MASO

Le module de communication permet aux agents de pouvoir interroger l'ontologie et recevoir les résultats venant d'elle. Donc, même pour un agent qui n'a aucun comportement envers les ontologies, ce module le rend capable de communiquer avec.

2.2.2 Communication entre plusieurs entités :

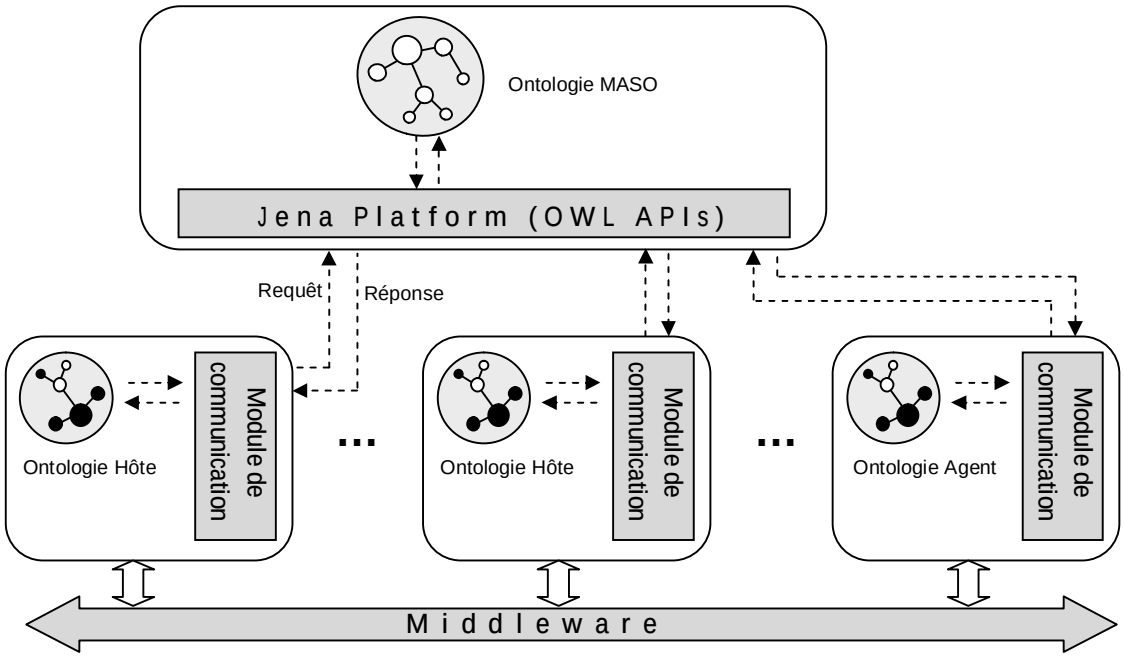

Figure 13 – Communication avec l'ontologie MASO dans un environnement dynamique

La Figure 13 montre comment plusieurs entités (agents et hôte), dans un environnement dynamique, peuvent utiliser l'ontologie MASO en ce qui concerne la sécurité du système.

Le module Plateforme Jena offre un certain nombre d'APIs OWL servant à exploiter l'ontologie.

2.3 Scénarios de communication :

Dans cette section, nous proposons deux scénarios de communication utilisant notre contribution (l'ontologie MASO) afin de montrer comment l'agent mobile se comporte dans un environnement dynamique envers la question de sécurité.

2.3.1 Scénario 1 :

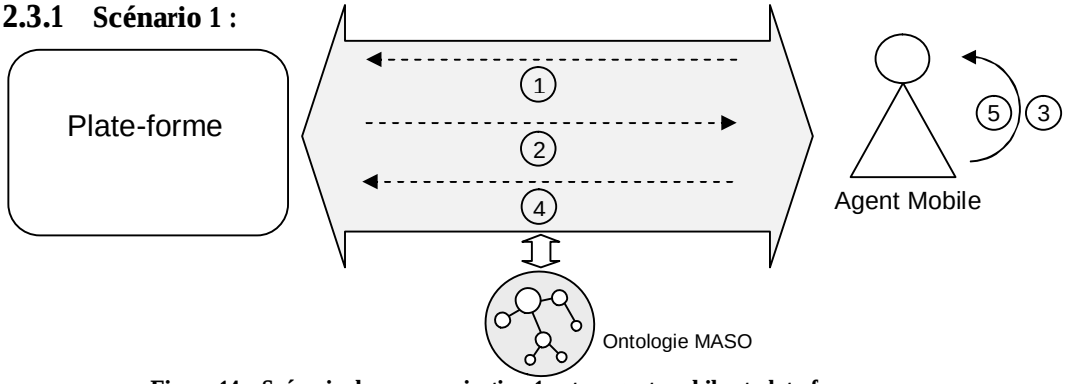

Figure 14 – Scénario de communication 1 entre agent mobile et plate-forme

(1) **Requête** : L'agent mobile envoie une requête à la plate-forme en demandant les paramètres de sécurité disponibles, et en fournissant les capacités dont il dispose.

(2) **Contexte** : La plate-forme, en utilisant l'ontologie MASO, renvoie à l'agent un contexte de sécurité contenant les différents paramètres de sécurité.

(3) **Raisonnement** : L'agent mobile, en utilisant des algorithmes dédiés, raisonne sur le contexte et vérifie les conditions exigées.

(4) **Notification** (exigences agent mobile $\not\subset$ capacités plate-forme) : Dans ce cas où le résultat de calcul (raisonnement) n'est pas satisfaisant, par exemple : un agent qui exige un protocole de sécurité non pris en charge par la plate-forme, l'agent mobile peut envoyer une notification expliquant son refus d'être exécuté. Cette notification sera stockée dans une base de données de la plate-forme afin d'être analyser plus tard, en cas de nécessité, pour améliorer les services de cette dernière.

(5) **Adaptation** : Si l'agent mobile a le pouvoir de s'adapter, en cas d'une hétérogénéité dans les paramètres de sécurité, le module de reconfiguration envoie la décision au noyau pour l'adaptation.

2.3.2 Scénario 2 :

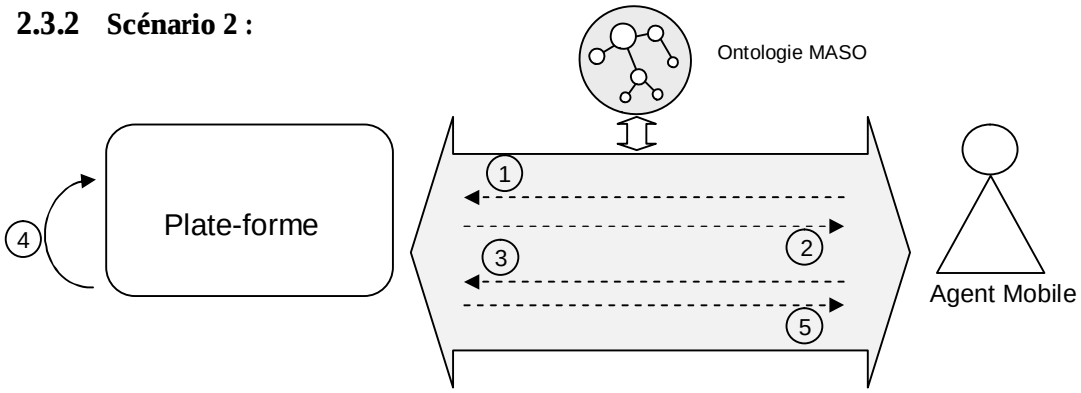

Figure 15 – Scénario de communication 2 entre agent mobile et plate-forme

(1) **Requête** : L'agent mobile envoie une requête à la plate-forme en demandant les paramètres de sécurité disponibles, et en fournissant les capacités dont il dispose.

(2) **Notification** : S'il n y a pas une confiance préétablie, la plate-forme réclame le contexte de sécurité de l'gent mobile en lui envoyant une demande.

(3) **Contexte** : L'agent mobile renvoie le contexte de sécurité en utilisant l'ontologie MASO pour éliminer toute hétérogénéité possible.

(4) **Raisonnement et analyse** : La plate-forme raisonne sur le contexte de l'agent mobile pour décider.

(5) **Notification** (exigences plate-forme $\not\subset$ capacité agent mobile) : En cas d'un manque de capacités d'agent mobile, la plate-forme envoie une notification pour que l'agent soit capable de comprendre la cause de l'échec.

3 PROCESSUS DE CONSTRUCTION DE L'ONTOLOGIE

Nous utilisons un processus de construction dans le développement de l'ontologie partant de connaissances brutes et arrivant à une ontologie d'application opérationnelle représentée par le langage OWL. Les grandes étapes de ce processus sont inspirées de la méthodologie de construction d'ontologies « METHONTOLOGY» [**Fernandez97**]. L'application de chacune des étapes de ce processus est basée sur l'exploitation du travaille de HEMMAM [**Hemam04**]. Ce processus est composé de cinq étapes :

- Spécification des besoins.
- Conceptualisation.
- Formalisation.
- Implémentation.
- Test & évolution de l'ontologie.

3.1 Spécification :

Cette étape consiste à établir un document formel de spécification des besoins représenté dans le langage RDF. Ce dernier permet de décrire l'ontologie à construire à travers les cinq aspects suivants :

- **Le domaine de connaissance** : déterminer aussi précisément que possible le domaine que va couvrir l'ontologie.
- **L'objectif** : le but de l'ontologie à créer pour le domaine considéré.
- **Les utilisateurs** : identifier au maximum les futurs utilisateurs de l'ontologie à créer.
- **Les sources d'informations** : déterminer les sources d'informations d'où les connaissances seront obtenues, par exemple, les experts du domaine, les documents techniques, etc.
- **La portée de l'ontologie** : déterminer à priori la liste des termes (les plus importants) pour le domaine à représenter.

3.2 Conceptualisation :

C'est l'étape la plus importante dans le processus de construction de l'ontologie. Elle est inspirée de la méthodologie METHONTOLOGY qui consiste à identifier et à structurer, à partir des sources d'informations, les connaissances du domaine. Elles permettent d'aboutir à un ensemble de représentations intermédiaires semi-formelles indépendamment des langages de formalisations à utiliser pour représenter l'ontologie. A la fin de cette phase, nous obtenons une ontologie conceptuelle.

Pour cela on distingue les principales tâches suivantes :

- Construction du glossaire de termes.
- Construction du diagramme de classification des concepts.
- Construction du diagramme de relations binaires.

- Construction du dictionnaire de concepts.
- Décrire les relations dans une table de relations binaires.
- Spécifier des contraintes sur les attributs dans une table d'attributs.
- Spécifier des axiomes sur les concepts dans une table d'axiomes logiques.
- Décrire les instances des concepts dans une table d'instances.

3.3 Formalisation :

Cette phase consiste à formaliser l'ontologie conceptuelle obtenue dans la phase précédente afin de faciliter sa représentation ultérieure dans un langage complètement formel et opérationnel. Notre choix est porté sur le formalisme de représentation est la logique de descriptions en s'appuyant sur la syntaxe de la logique de descriptions de type $SHOIN$ qui présente une logique de description très expressive et offre un ensemble de constructeurs riche pour décrire les concepts.

La logique de description est constituée de deux parties : une partie terminologique (TBOX) permettant de décrire les concepts et les rôles et d'une partie assertionnelle (ABOX) décrivant les instances.

3.4 Implémentation :

L'ontologie que nous avons obtenue dans la phase précédente est appelée une ontologie formelle. Le but de cette étape sera donc de coder l'ontologie formelle en OWL DL qui dispose de fonctionnalités sémantiques plus riches que ses prédécesseurs RDFS ou DAML+OIL. A la fin ce cette phase, nous obtenons une ontologie opérationnelle.

Afin de faciliter le processus de codification, nous utilisons PROTEGE OWL version 3.1.1 **[NOY00]** qui dispose d'une interface modulaire, développée au Stanford Medical Informatics de l'Université de Stanford, permettant l'édition, la visualisation et le contrôle d'ontologies, et contient des classes (concepts), des slots (propriétés) et des facettes (valeurs des propriétés et contraintes), ainsi que des instances des classes et des propriétés.

3.5 Tests et évolution

Cette étape consiste à exploiter les services d'inférence fournis par la logique de description afin de supporter le processus de construction et d'améliorer la qualité de l'ontologie. Pour ce faire, nous proposons l'utilisation de l'outil RACER, un système de la logique de descriptions. Ce dernier, permet de lire un document au format OWL (ontologie OWL) et de le représenter sous forme d'une base de connaissances LD et de fournir des services d'inférence pour les niveaux TBOX et ABOX.

Cette étape sert aussi à suivre l'évolution de l'ontologie, c'est-à-dire les nouveaux concepts à ajouter dans la partie terminologique (TBOX) de l'ontologie. Une classification a lieu chaque fois qu'une définition de concept est nouvellement créée. Le mécanisme de raisonnement de base des logiques de description est la classification de concepts. Elle est réalisée par un algorithme de classification, appelé «le classifieur». Le classifieur utilise la description d'un

nouveau concept pour le placer à l'endroit correspondant dans la hiérarchie. Afin de trouver la place appropriée au nouveau concept, l'algorithme de classification détermine les relations de subsomption entre ce concept et les autres; Ces relations peuvent être spécifiées directement, trouvées par transitivité ou calculées à partir de la sémantique des conditions des rôles. La recherche de la place correcte pour le nouveau concept comporte trois étapes :

1. La recherche des subsumants les plus spécifiques SPS (concepts qui subsument le concept à classer et dont les fils ne le subsument pas)
2. La recherche des subsumés les plus généraux SPG (concepts subsumés par le concept à classer et dont les pères ne sont pas subsumés par lui).
3. Insertion du nouveau concept dans la hiérarchie.

4 CONSTRUCTION D'UNE ONTOLOGIE POUR LE DOMAINE DE LA SÉCURITÉ

Dans cette section, nous construisons notre ontologie qui concerne le domaine de sécurité pour les agents mobiles. A cette fin, nous suivrons les étapes du processus de construction d'ontologie proposé dans la section **3**.

4.1 Spécification :

Le développement de l'ontologie est débuté par la phase de spécification qui consiste à établir un document de spécification des besoins. Au sein de ce document, nous dériverons l'ontologie à construire à travers les cinq aspects suivants :

- **Le domaine de connaissance :** l'ontologie, que nous venons de construire, s'inscrit dans le cadre de la sécurité des agents mobiles. En effet, elle peut prendre ses concepts depuis le domaine de sécurité informatique, et le domaine des agents mobiles. Pour le besoin de notre travail, on se contente des concepts qui caractérisent un contexte de sécurité clair de l'agent mobile et de l'hôte.
- **L'objectif :** l'objectif majeur de l'incorporation des ontologies au sein d'un système d'agents mobiles est de celer l'hétérogénéité concernant la sécurité entre les agents et les hôtes afin de garantir une meilleure interopérabilité des politiques de sécurité.
- **Les utilisateurs :** cet aspect présente l'ensemble des utilisateurs pouvant exploiter l'ontologie. Dans notre cas, les utilisateurs de l'ontologie représentent les agents mobiles et les hôtes qui doivent exploiter les ontologies afin d'atteindre l'objectif visé.
- **Les sources d'informations :** les sources d'informations sur lesquelles nous nous sommes basés pour arriver à la construction de l'ontologie d'applications sont des documents techniques de la sécurité des agents mobiles, la technologie de Web services, et des ontologies pour la sécurité informatique.
- **La portée de l'ontologie :** Cet aspect consiste à déterminer à priori la liste des termes de l'ontologie (les plus importants), parmi ces termes, nous pouvons citer : Agent, Hôte, Algorithme, Protocole, Contre-mesure, Menace, …etc.

Nous résumons cette phase dans un document RDF présenté dans la Figure 16. Il peut inclure aussi d'autres aspects tels que : la date de création de l'ontologie, ses créateurs, son niveau de formalité …etc.

```
<rdf:RDF>
<rdf: Description about=" URI of ontology" >
    <Domaine> sécurité pour les agents mobiles </ Domaine>
    <Date> 16 Décembre 2008 </ Date>
    <Développé-par>
    <rdf:Sequence>
    <rdf:_1 R. LEKHCHINE, laboratoire LIRE, Université Mentouri de Constantine >
    <rdf:_2 Z. BOUFAÏDA, laboratoire LIRE, Université Mentouri de Constantine >
    </rdf:Sequence>
    </ Développé-par >
    <Objectif> une ontologie modélisant les propriétés de la sécurité des agents mobiles, ainsi que les
    propriétés sur l'environnement d'exécution dans lequel les agents peuvent coopérer, l'objectif de la
    construction de cette ontologie est de fournir une sémantique claire, interprétable, et partagée entre les
    agents mobiles et les hôtes.
    </ Objectif>
    <Niveau de formalité> formel </ Niveau de formalité >
    <Termes>
    <rdf:Sequence>
    <rdf:_1 Agent> <rdf:_2 Hôte> <rdf:_3 Protocole > …
    </rdf:Sequence>
    </ Termes>
    <Sources>
    <rdf:Sequence> …
    </rdf:Sequence>
    </ Sources>
</rdf:description>
</rdf:RDF>
```

Figure 16 - Un document RDF de spécification de l'ontologie.

L'ontologie MASO est sensée fournir des réponses aux questions suivantes :

- Quels sont les différents protocoles de communications supportés ?
- Quelles sont les différentes menaces ?
- Quelles sont les exigences de sécurité ?
- Quelles sont les contre-mesures fournies ?
- Quelles sont les insuffisances ou les exigences manquées ?

4.2 Conceptualisation :

Une fois que la majorité des connaissances est acquise, on doit les organiser et les structurer en utilisant des représentations intermédiaires semi-formelles qui sont faciles à comprendre et sont indépendantes de tout langage d'implémentation. Cette phase comporte plusieurs étapes qui sont :

- Construction de glossaire de termes.
- Construction de diagramme de classification de concepts.
- Construction de diagramme de relations binaires.
- Dictionnaire de concepts.
- Tableaux des relations binaires.
- Tableaux des attributs.
- Tableaux des axiomes logiques.
- Tableaux des instances.

4.2.1 Construction de glossaire de termes :

Ce glossaire contient la définition de tous les termes relatifs au domaine (concepts, instances, attributs, relations) qui seront représentés dans l'ontologie finale, par exemple, dans notre cas les termes Agent et Algorithme sont des concepts, *assureObjectif* et *utiliseProtocole* représentent des relations,…etc.

Le Tableau 9 fournit une liste détaillée des différents termes utilisés dans l'ontologie. Nous avons ignoré, pour éviter une éventuelle répétition, les termes déjà définis tels que la confiance, la confidentialité, la contre-mesure … etc. :

Nom du terme	Synonymes	Acronymes	Description
Adresse	Adress	-	Valeur affectée à chaque station connectée sur l'Internet
Adresse-IP	IP	-	Adresse codée sur 32 bits selon le protocole Internet et affectée à un ordinateur figurant dans un réseau
Agent	-	-	Entité autonome capable de communiquer et se déplacer d'un site à un autre en cours d'exécution
Algorithme	-	-	Ensemble de prescriptions et de règles qui définissent « ce qu'il faut faire » et « dans quel ordre » pour résoudre un problème
Anonymat	Anonymity	-	Etat de ce qui est anonyme (inconnu)
Antivirus	-	-	Logiciel destiné à protéger un poste de travail contre les attaques de virus, lorsque celui-ci est connu
Asset	Ressource	-	Ce qu'on emploie pour accomplir une tâche, se tirer d'un embarras ou vaincre des difficultés
Assure-Objectif(Mécanisme, Objectif)	-	-	Un mécanisme assure un ou plusieurs objectifs
Attaque	Attack	-	Exploitation d'une faille d'un système informatique
Authentification	Identification	-	Procédé permettant de vérifier l'identité d'une entité, largement utilisé dans les réseaux informatiques
Autorisation	-	-	Désigne un attribut listant les permissions d'accès
Capable-Mécanisme (Entité, Mécanisme)	-	-	Une entité est capable d'utiliser un ou plusieurs mécanismes de sécurité
Capacité	Taille	-	Mesure le stockage dont un système est capable
Carte-de-crédit	-	-	Moyen de paiement universel permettant de payer chez les commerçants et d'effectuer des retraits aux distributeurs automatiques de billets
Certificat	-	-	Désigne l'identification d'un élément (équipement, société, personne), sous une forme électronique

Chemin	Path	-	Désigne le circuit emprunté dans une arborescence pour parvenir à un fichier
Cible (Menace, Asset)	-	-	Une Menace cible une ou plusieurs ressources (Asset)
Clé-privée	-	-	Une clé utilisée pour le chiffrement asymétrique
Clé-publique	-	-	Une clé utilisée pour le chiffrement asymétrique
Clé-secrète	-	-	Une clé utilisée pour le chiffrement symétrique
Composant	Component	-	Objet faisant partie d'un système
Contrôle-d'accès	Access-control	-	Dispositif qui valide les demandes d'accès à des données
Cookie	-	-	Fichier déposé par un serveur dans l'ordinateur qui le visite et qui rend possible l'identification des machines
Data	Données	-	Informations codées et lisibles par la machine
Date-expiration	-		Date après laquelle un droit d´option ne peut plus être exercé
Défaut	Fault	-	Imperfection matériel ou logiciel
Description	-	-	Ensemble d'information sur une personne, un paysage, une activité en vue d'en avoir une maîtrise et une connaissance parfaite
Dispose (Hôte, Asset)	-	-	Un hôte dispose un ou plusieurs ressources
Entité	-	-	Abstraction considérée comme une réalité
Erreur	Error	-	Evènement traduisant le dysfonctionnement d'un système
Exige-Objectif (Entité, Objectif)	-	-	Une entité exige un ou plusieurs objectifs
Exploite (Menace, Vulnérabilité)	-	-	Une menace exploite une ou plusieurs vulnérabilités
Fax	-	-	Numéro de fax de propriétaire d'agent mobile
Firewall	Pare-feu	-	Logiciel ou matériel permettant de protéger les éléments d'un réseau des attaques d'un autre réseau lié par une passerelle
Hardware	Matériel	-	Matériel informatique physique, par opposition au software
Hôte	Host	-	Ordinateur qui dispose des ressources particulières, et qui est connecté à un réseau
Initialise (Entité, Menace)	-	-	Une entité peut initialiser une ou plusieurs menaces
Issuer	-	-	Organisme publicateur
Longueur-minimum	Minlength	-	Nombre de bits minimum représentant la taille de la clé
Marque	-	-	Signe distinctif qui permet au consommateur de distinguer le produit ou service d'une entreprise de ceux proposés par les entreprises concurrentes

Mémoire	Memory	RAM	Bloc principal de mémoire vive d'une machine à l'intérieur duquel tous les logiciels d'application se chargent et s'exécutent
Menace	Threat	-	Manifestation de violence par laquelle on signifie à autrui l'intention que l'on a de faire du mal
Microprocesseur	Processeur	CPU	Pièce de silicium autour de laquelle est articulé un ordinateur
Mot-de-passe	Password, Code	-	Séquence de caractères utilisée par un usager pour valider son accès à des ressources personnelles
Nom	Name	-	Mot qui sert à désigner une personne, un organisme ou une chose
Not-After	-	-	Date limite-fin de validité d'un certificat X.509
Not-before	-	-	Date limite-début de validité d'un certificat X.509
Numéro-de-série	Serial-number	NS	Un numéro de série est un nombre unique qui est assigné à un objet d'une série afin de l'identifier
Passeport	Credential	-	Ce qui donne droit à la confiance, de crédit, ou de l'autorité
Prénom	-	-	Nom précédent le nom de famille
Propriétaire	Owner	-	Qui possède quelque chose
Propriété-de (Agent, Propriétaire)	-	-	Un agent est la propriété d'un seul propriétaire
Protection	-	-	Dispositif de sécurité isolant les opérateurs d'un danger potentiel
Protège (Contre-mesure, Asset)	-	-	Une contre-mesure protège une ou plusieurs ressources
Protocole	-	-	Ensemble des conventions nécessaires pour faire coopérer des entités distantes, en particulier pour établir et entretenir des échanges d'informations entre ces entités
Réduit (Contre-mesure, Vulnérabilité)	-	-	Une contre-mesure réduit une ou plusieurs vulnérabilités
Service	-	-	Composant implémenté dans n'importe quel langage, déployé sur n'importe quelle plate-forme. Il doit pouvoir être découvert et invoqué dynamiquement par d'autres services
Société	Entreprise	-	La société est instituée par deux ou plusieurs personnes qui conviennent par un contrat d'affecter à une entreprise commune des biens ou leur industrie en vue de partager le bénéfice ou de profiter de l'économie qui pourra en résulter
Software	Logiciel	-	Par opposition à hardware, un software est un logiciel

Storage	Mémoire-secondaire, Mémoire-de-masse	-	Périphérique de stockage
Téléphone	-	Tel	Nombres désignant le numéro de téléphone d'une personne ou organisme
Token	-	-	Moyen utilisé dans le processus d'identification/authentification
Trojan-Horse	Cheval-de-Troie	-	Un cheval de Troie est un programme installé discrètement par un pirate sur un ordinateur, simulant une certaine action, mais faisant tout autre chose en réalité.
Utilise-Algorithme (Protocole, Algorithme)	-	-	Un protocole utilise un ou plusieurs algorithmes
Utilise-Passeport (Algorithme, Passeport)	-	-	Un algorithme utilise un ou plusieurs passeports
Utilise-Protocole (Mécanisme, protocole)	-	-	Un mécanisme utilise un ou plusieurs protocoles
Valeur	Value	-	Valeur sauvegardée par le cookie
Version	-	-	La version d'un certificat numérique
Virus	-	-	Morceau de programme informatique malicieux conçu et écrit pour exécuter des opérations malveillantes
Vitesse	Speed	-	Exprimée en gigahertz (GHz) à présent, la fréquence du processeur désigne le nombre d'opérations effectuées en une seconde par le processeur
Worm	Ver	-	Un Ver est un petit programme qui se copie d'ordinateur en ordinateur. La différence entre un ver et un virus est que le ver ne peut pas se greffer à un autre programme et ne peut donc l'infecter.
X.509	-	-	Norme de cryptographie de l'Union internationale des télécommunications pour les infrastructures à clés publiques

Tableau 9 – Glossaire de termes

4.2.2 Construction du diagramme de classification de concepts :

Dans cette étape, nous construisons le diagramme de classification de concepts. La hiérarchie de classification de concepts démontre l'organisation des concepts de l'ontologie en un ordre hiérarchique qui exprime les relations sous classe.

Un concept universel « **Thing** », qui généralise tous les concepts racines des différentes hiérarchies de concepts est utilisé pour former une seule hiérarchie globale.

Pour construire la taxonomie des concepts, METHONTOLOGY propose à utiliser les quatre relations : *Subclass-Of, Disjoint-Decomposition, Exhaustive-Decomposition,* et *Partition.*

- Un concept C1 est une *sous-classe* de concept C2 si et seulement si toute instance de C1 est une instance de C2. Par exemple, Agent est une sous-classe de la classe Entité.
- Une *Disjoint-Decomposition* d'un concept C est un ensemble de sous-classes de C qui ne couvrent pas C et n'ont pas des instances communes. Par exemple, Les concepts RBAC et X.509 forment une *Disjoint-Decomposition* de concept Certificat.
- Une *Exhaustive-Decomposition* d'un concept C est un ensemble de sous-classes de C qui couvrent C et peuvent avoir des instances communes.
- Une *Partition* d'un concept C est un ensemble de sous-classes de C qui couvrent C et n'ont aucune instance commune. Par exemple, Les concept Agent et Hôte forment une *Partition* de concept Entité.

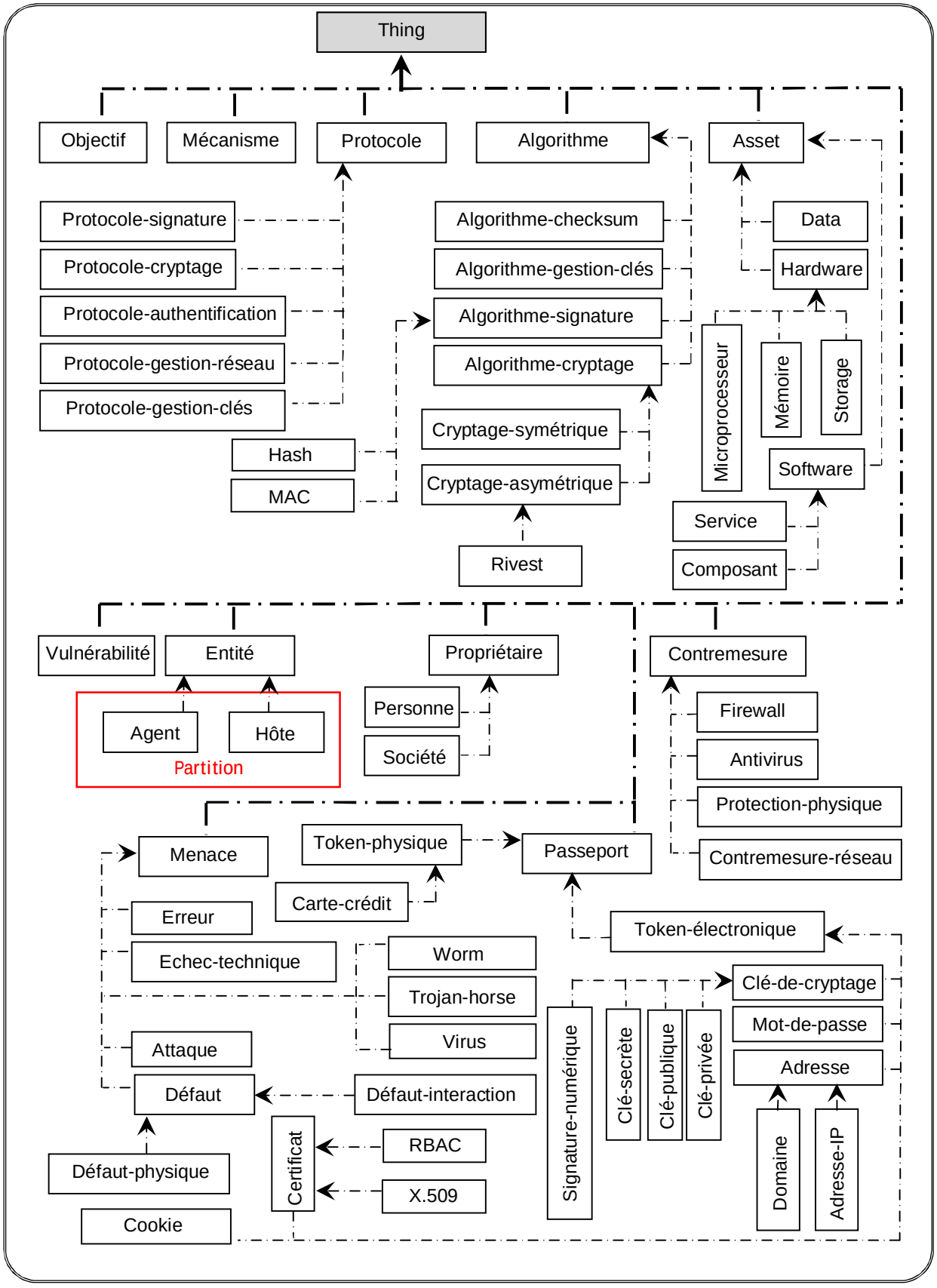

Figure 17 – Diagramme de classification de concepts

4.2.3 Construction de diagramme de relations binaires :

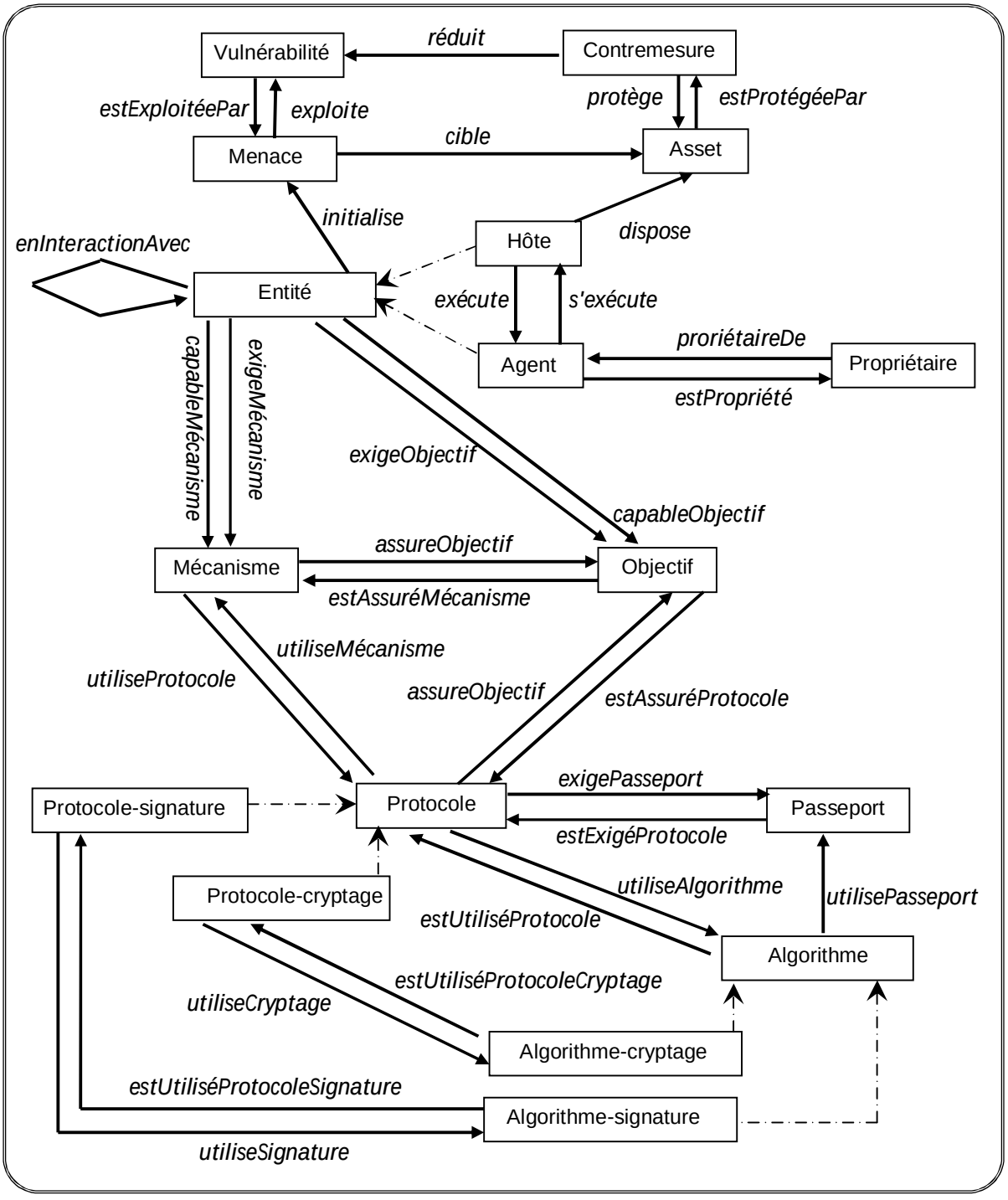

Figure 18 – Diagramme de relations binaires

4.2.4 Dictionnaire de concepts :

Nom de concept	Instances	Attributs de classe	Attributs d'instance	Relations
Adresse	-	-	-	-
Adresse-IP	-	-	-	-
Agent	-	-	-	estPropriété sexécute
Algorithme	-	-	-	utilisePasseport estUtiliséProtocole
Algorithme-checksum	-	-	-	-
Algorithme-cryptage	-	-	-	estUtiliséProtocoleCryptage
Algorithme-gestion-clés	-	-	-	-
Algorithme-signature	SHA1 SHA256 MD5	-	-	estUtiliséProtocoleSignature
Antivirus	-	-	nom version	-
Asset	-	-	-	estProtégéePar
Attaque	-	-	-	-
Carte-crédit	-	-	date-expiration	-
Certificat	-	-	-	-
Clé-de-cryptage	-	-	longueur-clé	-
Clé-privée	-	-	-	-
Clé-publique	-	-	-	-
Clé-secrète	-	-	-	-
Composant	-	-	-	-
Contremesure	-	-	-	réduit protège
Contremesure-réseau	-	-	-	-
Cookie	-	-	nom valeur-cookie chemin	-
Cryptage-asymétrique	RSA ElGamal EllipticCurve	-	-	-
Cryptage-symétrique	AES DES IDEA 3DES BlowFish TwoFish CAST Skipjack	-	-	-
Data	-	-	-	-
Défaut	-	-	-	-
Défaut-interaction	-	-	-	-

Défaut-physique	-	-	-	-
Domaine	-	-	-	-
Echec-technique	-	-	-	-
Entité	-	-	nom-entité	Initialise capableObjectif exigeObjectif exigeMécanisme capableMécanisme enIntéractionAvec
Erreur	-	-	-	-
Firewall	-	-	-	-
Hardware	-	-	-	-
Hash	-	-	-	-
Hôte	-	-	-	dispose exécute
MAC	CBCMAC HMAC	-	-	-
Mécanisme	Authentification Identification Autorisation Gestion-clés Contrôle-d'accès	-	-	utiliseProtocole assureObjectif
Mémoire	-	-	capacité	-
Menace	-	-	-	cible exploite
Microprocesseur	-	-	marque modèle vitesse	-
Mot-de-passe	-	-	longueur-minimum	-
Objectif	Disponibilité Confidentialité Intégrité Non-répudiation Confiance Anonymat	-	-	estAssuréProtocole estAssuréMécanisme
Passeport	-	-	-	estExigéProtocole
Personne	-	-	Prénom	-
Propriétaire	-	-	nom Tel Fax e-mail	propriétaireDe
Protection-physique	-	-	-	-
Protocole	-	-	-	assureObjectif exigePasseport utiliseAlgorithme
Protocole-authentification	-	-	-	-
Protocole-cryptage	-	-	-	utiliseCryptage

Protocole-gestion-clés	-	-	-	-
Protocole-gestion-réseau	SSL SSH	-	-	-
Protocole-signature	-	-	-	utiliseSignature
RBAC	-	-	-	-
Rivest	RC2 RC4 RC5 RC6	-	-	-
Service	-	-	-	-
Signature-numérique	-	-	-	-
Société	-	-	-	-
Software	-	-	nom description-logiciel	-
Storage	-	-	capacité	-
Token-électronique	-	-	-	-
Token-physique	-	-	date-expiration	-
Trojan-horse	-	-	-	-
Virus	-	-	nom	-
Vulnérabilité	-	-	-	estExploitéePar
Worm	-	-	-	-
X.509	-	-	numéro-de-série issuer notBefore notAfter version	-

Tableau 10 – Dictionnaire de concepts

4.2.5 Tableaux des relations binaires :

Nom de relation	Concept source	Cardinalité source (max)	Concept cible	Relation inverse
assureObjectif	Mécanisme	N	Objectif	estAssuréMécanisme
assureObjectif	Protocole	N	Objectif	estAssuréProtocole
capableMécanisme	Entité	N	Mécanisme	-
capableObjectif	Entité	N	Objectif	-
cible	Menace	N	Asset	-
dispose	Hôte	N	Asset	-
enInteractionAvec	Entité	N	Entité	enInteractionAvec
estAssuréMécanisme	Objectif	N	Mécanisme	-
estAssuréProtocole	Objectif	N	Protocole	-
estExigéProtocole	Passeport	N	Protocole	exigePasseport
estExploitéPar	Vulnérabilité	N	Menace	exploite
estPropriété	Agent	1	Propriétaire	propriétaireDe
estProtégéePar	Asset	N	Contremesure	protège
estUtiliséProtocole	Algorithme	N	Protocole	utiliseAlgorithme
estUtiliséProtocoleCryptage	Algorithme-cryptage	N	Protocole-cryptage	utiliseCryptage
estUtiliséProtocoleSignature	Algorithme-signature	N	Protocole-signature	utiliseSignature
exécute	Hôte	N	Agent	s'exécute
exigeMécanisme	Entité	N	Mécanisme	-
exigeObjectif	Entité	N	Objectif	-
exigePasseport	Protocole	N	Passeport	estExigéProtocole
exploite	Menace	N	Vulnérabilité	estExploitéPar
initialise	Entité	N	Menace	-
propriétaireDe	Propriétaire	N	Agent	estPropriété
protège	Contremesure	N	Asset	estProtégéePar
réduit	Contremesure	N	Vulnérabilité	-
s'exécute	Agent	N	Hôte	exécute
utiliseAlgorithme	Protocole	N	Algorithme	estUtiliséProtocole
utiliseCryptage	Protocole-cryptage	N	Algorithme-cryptage	estUtiliséProtocoleCryptage
utiliseMécanisme	Protocole	N	Mécanisme	utiliseProtocole
utilisePasseport	Algorithme	N	Passeport	-
utiliseProtocole	Mécanisme	N	Protocole	utiliseMécanisme
utiliseSignature	Protocole-signature	N	Algorithme-signature	estUtiliséProtocoleSignature

Tableau 11 – Tableau des relations binaires

4.2.6 Tableaux des attributs :

Nom d'attribut d'instance	Nom de concept	Type de valeur	Intervalle de valeur	Cardinalité
capacité	Storage	entier	-	(1, 1)
capacité	Mémoire	entier	-	(1 ,1)
chemin	Cookie	string	-	(1, 1)
date-expiration	Carte-crédit	date	-	(1, 1)
description-logiciel	Software	string	-	(0, N)
e-mail	propriétaire	string	-	(0, N)
fax	propriétaire	string	-	(0, N)
issuer	X.509	string	-	(1, 1)
longueur-clé	Clé-de-cryptage	entier	>0	(1, 1)
longueur-minimum	Mot-de-passe	entier	-	(1, 1)
marque	Microprocesseur	string	-	(1, 1)
modèle	Microprocesseur	string	-	(1, 1)
nom	Software	string	-	(1, 1)
nom	Antivirus	string	-	(1, 1)
nom	Cookie	string	-	(1, 1)
nom	propriétaire	string	-	(1, 1)
nom	Virus	string	-	(1, 1)
notAfter	X.509	date	-	(1, 1)
notBefore	X.509	date	-	(1, 1)
numéro-de-série	X.509	string	-	(1, 1)
prénom	personne	string	-	(1, 1)
téléphone	propriétaire	string	-	(0, N)
valeur-cookie	Cookie	entier	-	(1, 1)
version	X.509	string	-	(1, 1)
version	Antivirus	string	-	(1, 1)
vitesse	Microprocesseur	entier	-	(1, 1)

Tableau 12 – Tableau des attributs

4.2.7 Tableaux des axiomes logiques :

Description	Expression	Concepts concernés	Relations concernées
Chaque entité est soit un agent, soit un hôte.	$\forall X: Entité(X) \Rightarrow Agent(X) \vee Hôte(X)$	Entité Agent Hôte	-
Les concepts agent et hôte sont incompatibles.	$(\forall X: Agent(X) \Rightarrow not\ Hôte(X)) \wedge (\forall Y: Hôte(Y) \Rightarrow not\ Agent(Y))$	Agent Hôte	-
Chaque algorithme de cryptage est soit symétrique, soit asymétrique.	$\forall X: Algorithme\_cryptage(X) \Rightarrow Cryptage\_symétrique(X) \vee Cryptage\_asymétrique(X)$	Algorithme-cryptage Cryptage-symétrique Cryptage-asymétrique	-
Une entité ne peut pas être en interaction avec elle-même.	$\forall X, Y: Entité(X) \wedge Entité(Y) \wedge enInteractionAvec(X, Y) \Rightarrow X \neq Y$	Entité	enInteractionAvec

Tableau 13 – Tableau des axiomes logiques

4.2.8 Tableaux des instances :

Nom de l'instance	Nom du concept	Attributs	Valeurs
3DES	Cryptage-symétrique	-	-
AES	Cryptage-symétrique	-	-
Anonymat	Objectif	-	-
Authentification	Mécanisme	-	-
Autorisation	Mécanisme	-	-
BlowFish	Cryptage-symétrique	-	-
CAST	Cryptage-symétrique	-	-
CBCMAC	MAC	-	-
Confiance	Objectif	-	-
Confidentialité	Objectif	-	-
Contrôle-d'accès	Mécanisme	-	-
Disponibilité	Objectif	-	-
ElGamal	Cryptage-asymétrique	-	-
EllipticCurve	Cryptage-asymétrique	-	-
Gestion-clés	Mécanisme	-	-
HMAC	MAC	-	-
IDEA	Cryptage-symétrique	-	-
Identification	Mécanisme	-	-
Intégrité	Objectif	-	-
MD5	Algorithme-signature	-	-
Non-répudiation	Objectif	-	-
Rivest	RC2	-	-
Rivest	RC4	-	-
Rivest	RC5	-	-
Rivest	RC6	-	-
RSA	Cryptage-asymétrique	-	-
SHA1	Algorithme-signature	-	-
SHA256	Algorithme-signature	-	-
Skipjack	Cryptage-symétrique	-	-
SSH	Protocole-gestion-réseau	-	-
SSL	Protocole-gestion-réseau	-	-
TwoFish	Cryptage-symétrique	-	-

Tableau 14 – Tableau des instances

4.3 Formalisation :

Dans cette étape, nous allons utiliser le formalisme des logiques de descriptions afin de formaliser le modèle conceptuel que nous avons obtenu dans l'étape précédente de conceptualisation.

4.3.1 Construction de TBox :

Nous construisons la TBox en définissant des concepts et les rôles et en utilisant les constructeurs fournis par les logiques de descriptions. Par exemple, la définition « un propriétaire d'agent doit avoir au moins un agent » peut être écrite en logique de description par :

$$\text{Propriétaire} \equiv \exists \textit{propriétaireDe}$$

De plus, nous construisons la TBox par la spécification des relations de subsomption qui existent entre les différents concepts/rôles ; par exemple pour spécifier que la classe Personne est subsumée par la classe Propriétaire on écrit :

$$\text{Personne} \sqsubseteq \text{Propriétaire}$$

Les définitions de différents concepts sont illustrées dans le tableau ci-dessous.

Concept	Définition	Relations de subsomption
Adresse	-	Adresse ⊑ Token-électronique
AdresseIP	-	AdresseIP ⊑ Adresse
Agent	≡ (Entité ⊔ ¬Hôte) ⊓ =1*estPropriété*.Propriétaire	Agent ⊑ Entité
Algorithme	≡ ∃*utilisePasseport*.Passeport ⊓ ∃*estUtiliséProtocole*.Protocole	Algorithme ⊑ ⊤
Algorithme-checksum	-	Algorithme-checksum ⊑ Algorithme
Algorithme-cryptage	≡ ∃*estUtiliséProtocoleCryptage*.Protocole-cryptage	Algorithme-cryptage ⊑ Algorithme
Algorithme-gestion-clé	-	Algorithme-gestion-clés ⊑ Algorithme
Algorithme-signature	≡ ∃*estUtiliséProtocoleSignature*.Protocole-signature	Algorithme-signature ⊑ Algorithme
Antivirus	≡ =1 *nom*.String ⊓ =1 *version*.String	Antivirus ⊑ Contremesure
Asset	-	Asset ⊑ ⊤
Attaque	-	Attaque ⊑ Menace
Carte-crédit	≡ =1 *date-expiration*.Date	Carte-crédit ⊑ Token-physique
Certificat	-	Certificat ⊑ Token-électronique

Clé-de-cryptage	≡ =1 *longueur-clé*.Int	Clé-cryptée ⊑ Token-électronique
Clé-privée	-	Clé-privée ⊑ Clé-de-cryptage
Clé-publique	-	Clé-publique ⊑ Clé-de-cryptage
Clé-secrète	-	Clé-secrète ⊑ Clé-de-cryptage
Composant	-	Composant ⊑ Software
Contremesure	≡ ∃*protège*.Asset	Contremesure ⊑ ⊤
Contremesure réseau	-	Contremesure-réseau⊑ Contremesure
Cookie	≡ =1 *nom*.String ⊓ =1 *valeur-cookie*.Int ⊓ =1 *chemin*.String	Cookie ⊑ Token-électronique
Cryptage-asymétrique	-	Cryptage-asymétrique ⊑ Algorithme-cryptage
Cryptage-symétrique	-	Cryptage-symétrique ⊑ Algorithme-cryptage
Data	-	Data ⊑ Asset
Défaut	-	Défaut ⊑ Menace
Défaut-interaction	-	Défaut-interaction ⊑ Défaut
Défaut-physique	-	Défaut-physique ⊑ Défaut
Domaine	-	Domaine ⊑ Adresse
Echec-technique	-	Echec-technique ⊑ Menace
Entité	≡ (Agent ⊔ Hôte) ⊓ ∃*initialise*.Menace ⊓ ∃*exigeObjectif*.Objectif ⊓ ∃*capableObjectif*.Objectif ⊓ ∃*exigeMécanisme*.Mécansime ⊓ ∃*capableMécanisme*.Mécanisme	Entité ⊑ ⊤
Erreur	-	Erreur ⊑ Menace
Firewall	-	Firewall ⊑ Contremesure
Hardware	-	Hardware ⊑ Asset
Hash	-	Hash ⊑ Algorithme-signature
Hôte	≡ Entité ⊔ ¬Agent	Hôte ⊑ Entité
MAC	-	MAC ⊑ Algorithme-signature
Mécanisme	≡ {Autorisation, Identification, Authentification, Gestion-clés, Contrôle-daccès}	Mécanisme ⊑ ⊤
Mémoire	≡ =1 *capacité*.Int	Mémoire ⊑ Hardware

Menace	-	Menace ⊑ ⊤
Microprocesseur	≡ =1 *marque*.String ⊓ =1 *modèle*.String ⊓ =1 *vitesse*.Int	Microprocesseur ⊑ Hardware
Mot-de-passe	≡ =1 *longueur-minimum*.Int	Mot-de-passe ⊑ Token-électronique
Objectif	≡ {Confiance, Imputabilité, Integrité, Confidentialité, Non-répudiation, Disponibilité}	Objectif ⊑ ⊤
Passeport	-	Passeport ⊑ ⊤
Personne	≡ =1 *prénom*.String	Personne ⊑ Propriétaire
Propriétaire	≡ ∃*propriétaireDe*.Agent ⊓ =1 *nom*.String	Propriétaire ⊑ ⊤
Protection-physique	-	Protection-physique ⊑ Contremesure
Protocole	≡ ∃*assureObjectif*.Objectif ⊓ ∃*utiliseMécanisme*.Mécanisme ⊓ ∃*exigePasseport*.Passeport ⊓ ∃*utiliseAlgorithme*.Algorithme	Protocole ⊑ ⊤
Protocole-authentification	≡ *assureObjectif* : Confidentialité ⊔ *utiliseMécanisme* : Authentification	Protocole-authentification ⊑ Protocole
Protocole-cryptage	≡ ∃*utiliseCryptage*.Algorithme-cryptage	Protocole-crypatge ⊑ Protocole
Protocole-gestion-clés	≡ *utiliseMécanisme* : Gestion-réseau	Protocole-gestion-clés ⊑ Protocole
Protocole-gestion-réseau	-	Protocole-gestion-réseau ⊑ Protocole
Protocole-signature	≡ ∃*utiliseSignature*.Algorithme-signature	Protocole-signature ⊑ Protocole
RBAC	-	RBAC ⊑ Certificat
Rivest	-	Rivest ⊑ Cryptage-asymétrique
Service	-	Service ⊑ Software
Signature-numérique	-	Signature-numérique ⊑ Clé-cryptée
Société	-	Société ⊑ Propriétaire
Software	≡ =1 *nom*.String ⊓ ∃*description-logiciel*.String	Software ⊑ Asset
Storage	≡ =1 *capacité*.Int	Storage ⊑ Hardware
Token-électronique	-	Token-électronique ⊑ Passeport
Token-physique	-	Token-physique ⊑ Passeport
Trojan-horse	-	Trojan-horse ⊑ Menace
Virus	≡ =1 *nom*.String	Virus ⊑ Menace

Vulnérabilité	-	Vulnérabilité ⊑ ⊤
Worm	-	Worm ⊑ Menace
X.509	≡ =1 *numéro-de-série*.Int ⊓ =1 *issuer*.String ⊓ =1 *notBefore*.Date ⊓ =1 *notAfter*.Date ⊓ =1 version.String	X.509 ⊑ Certificat
Rôle	Relations de subsomption	
estUtiliséProtocoleCryptage	estUtiliséProtocoleCryptage ⊑ estUtiliséProtocole	
estUtiliséProtocoleSignature	estUtiliséProtocoleSignature ⊑ estUtiliséProtocole	
utiliseCryptage	utiliseCryptage ⊑ utiliseAlgorithme	
utiliseSignature	utiliseSignature ⊑ utiliseAlgorithme	

Tableau 15 – définition de la TBox

4.3.2 Construction de ABox :

Nous décrivons les faits en utilisant le langage assertionnel, de la manière suivante :

A(C) : Pour spécifier que A est une instance de la classe C
Par exemple : Confidentialité(Objectif).

R(A1, A2) : Pour spécifier que les deux individus A1 et A2 sont reliés par la relation R.
Par exemple : *assureObjectif*(SSL, Confidentialité).

Dans les deux tableaux Tableau 16, Tableau 17, nous définissons quelques assertions :

Concept	Définition
Objectif	Disponibilité(Objectif) Confidentialité(Objectif) Intégrité(Objectif) Non-répudiation(Objectif) Confiance(Objectif) Anonymat(Objectif)
Mécanisme	Authentification(Mécanisme) Identification(Mécanisme) Autorisation(Mécanisme) Gestion-clés(Mécanisme) Contrôle-d'accès(Mécanisme)
Algorithme-signature	SHA1(Algorithme-signature) SHA256(Algorithme-signature) MD5(Algorithme-signature)
Cryptage-asymétrique	RSA(Cryptage-symétrique) ElGamal(Cryptage-symétrique) EllipticCurve(Cryptage-symétrique)
Cryptage-symétrique	AES(Cryptage-symétrique) IDEA(Cryptage-symétrique) 3DES(Cryptage-symétrique) BlowFish(Cryptage-symétrique) TwoFish(Cryptage-symétrique) CAST(Cryptage-symétrique)

	Skipjack(Cryptage-symétrique)
MAC	CBCMAC(MAC) HMAC(MAC)
Rivest	RC2(Rivest) RC4(Rivest) RC5(Rivest) RC6(Rivest)
Protocole-gestion-réseau	SSL(Protocole-gestion-réseau) SSH(Protocole-gestion-réseau)

Tableau 16 – Partie assertionnelle des concepts

Relation	Définition
assureObjectif	assureObjectif(SSL, Confidentialité)
assureObjectif	assureObjectif(Authentification, Confidentialité)
estAssuréMécanisme	estAssuréMécanisme(Confiance, Identification)
assureObjectif	assureObjectif(Contrôle-d'accès, Disponibilité)

Tableau 17 – Partie assertionnelle des relations

4.4 Implémentation :

Après la conception, nous allons implémenter notre ontologie. Notre choix porte sur OWL qui représente un langage de codification utilisé pour implémenter l'ontologie en OWL, et cela pour toutes les fonctionnalités sémantiques que permet OWL et qui sont plus riches que celles des langages RDFS & DAML+OIL.

4.4.1 Présentation de l'éditeur PROTEGE OWL :

PROTEGE OWL **[ht4]** est une interface modulaire, développée au Stanford Medical Informatics de l'Université de Stanford7, permettant l'édition, la visualisation, le contrôle, l'extraction à partir de sources textuelles, et la fusion semi-automatique d'ontologies **[Noy00]**. PROTEGE OWL autorise la définition de méta-classes, dont les instances sont des classes, ce qui permet de créer son propre modèle de connaissances avant de bâtir une ontologie. De nombreux plug-ins sont disponibles ou peuvent être ajoutés par l'utilisateur. La Figure 19 présente l'interface de PROTEGE OWL version 3.4.

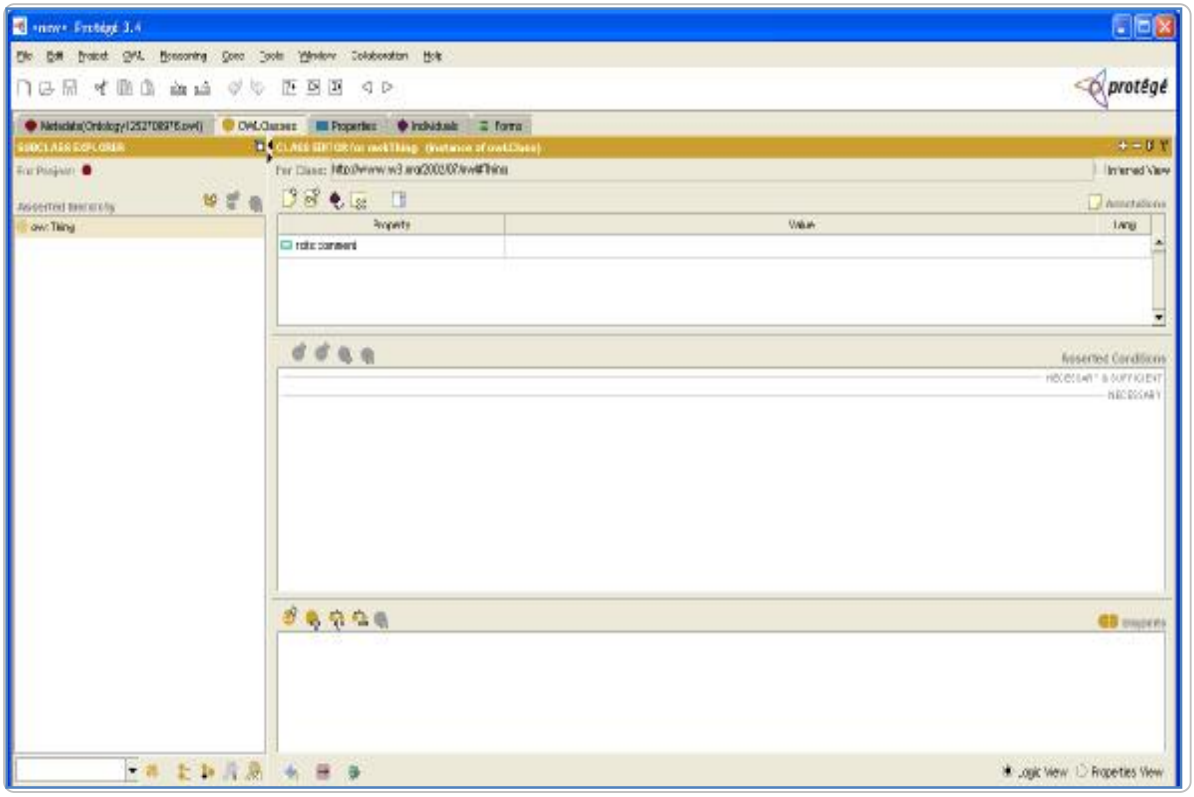

Figure 19 – Interface de Protégé OWL

L'interface, très bien conçue, et l'architecture logicielle permettant l'insertion de plug-ins pouvant apporter de nouvelles fonctionnalités (par exemple, la possibilité d'importer et d'exporter les ontologies construites dans divers langages opérationnels de représentation tels que OWL ou encore la spécification d'axiomes) ont participé au succès de PROTEGE OWL, qui regroupe une communauté d'utilisateurs très importantes et constitue une référence pour beaucoup d'autres outils **[Troncy04]**.

4.4.2 Définition de la hiérarchie des classes :

Nous commencerons tout d'abord par la création des concepts spécifiés dans l'étape de conceptualisation. PROTEGE OWL nous offre également un moyen de construire la hiérarchie de concepts, la Figure 20 présente comment-on procède pour créer un concept.

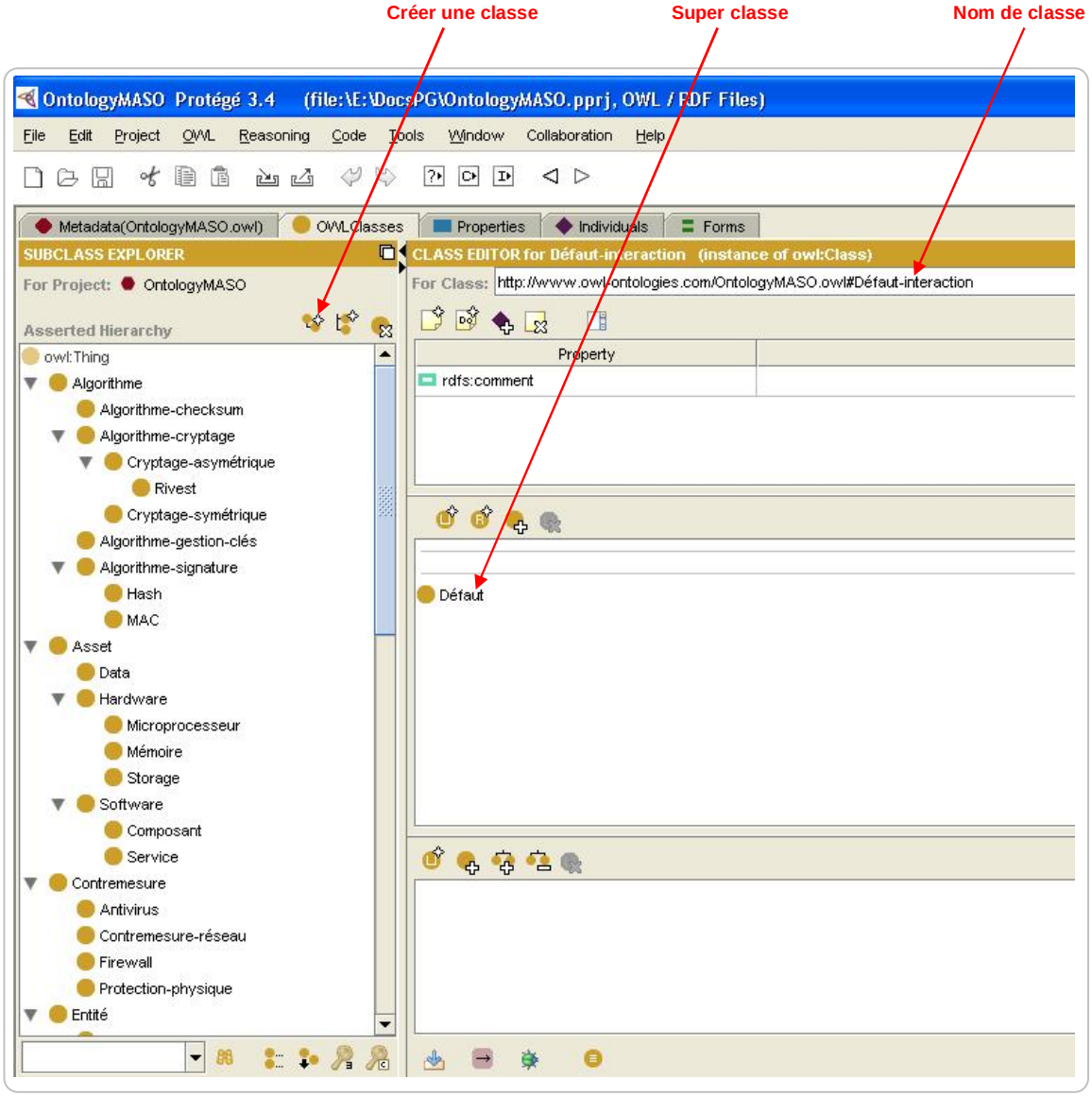

Figure 20 – Création des classes

4.4.3 Définition des propriétés :

Après avoir construit les classes, nous allons maintenant créer les propriétés pour chacun d'eux, les attributs vont être créés sous PROTEGE OWL par '*dataTypeProperty*' et les relations par '*objectProperty*'.

La Figure 21 montre les potentialités de PROTEGE OWL pour la création des propriétés.

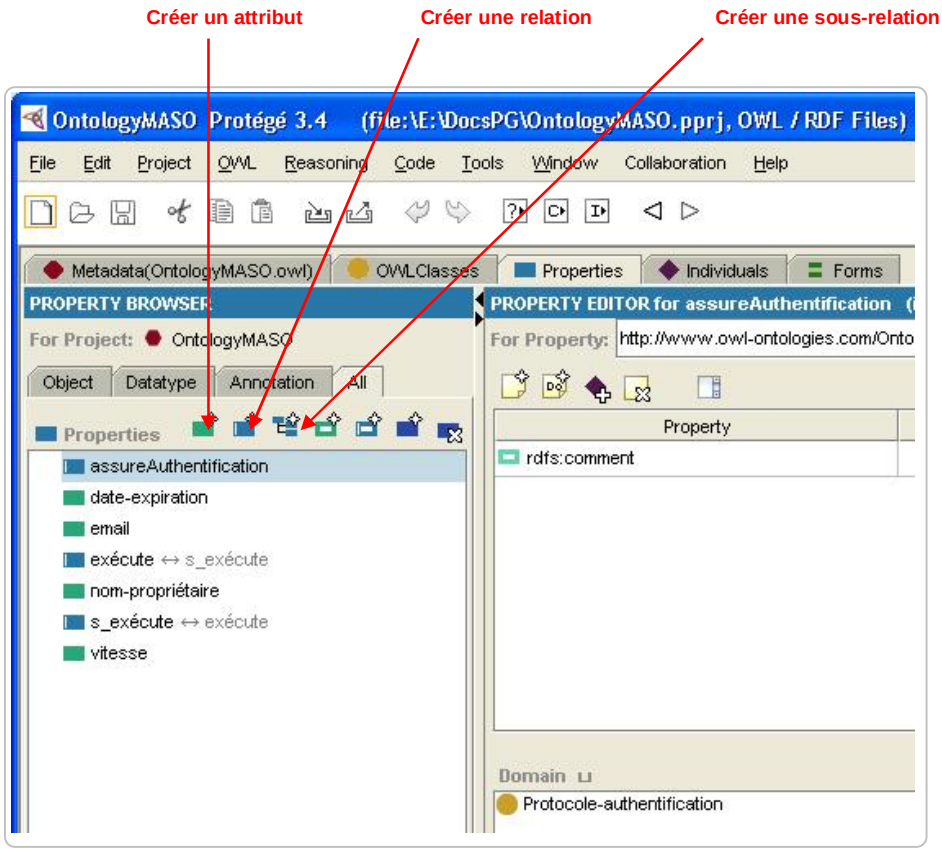

Figure 21 – Création des propriétés pour une classe

La Figure 22 montre comment créer un attribut et spécifier son nom, domaine et type.

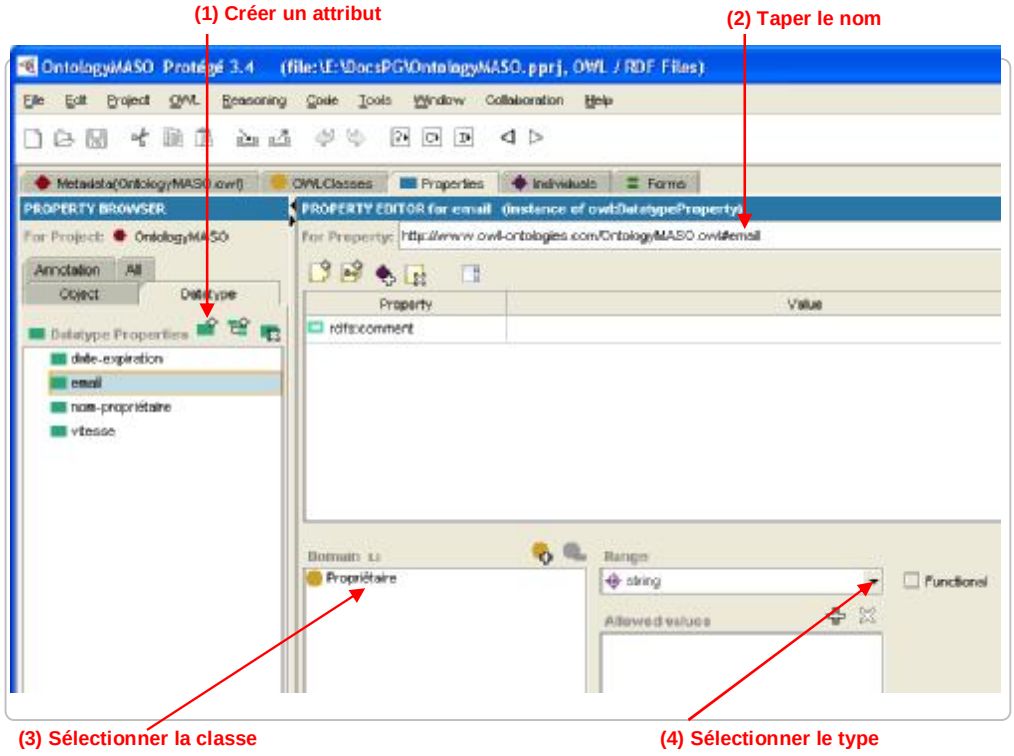

Figure 22 – Création d'un attribut

La Figure 23 montre comment créer une relation et spécifier son nom, type, domaine et co-domaine.

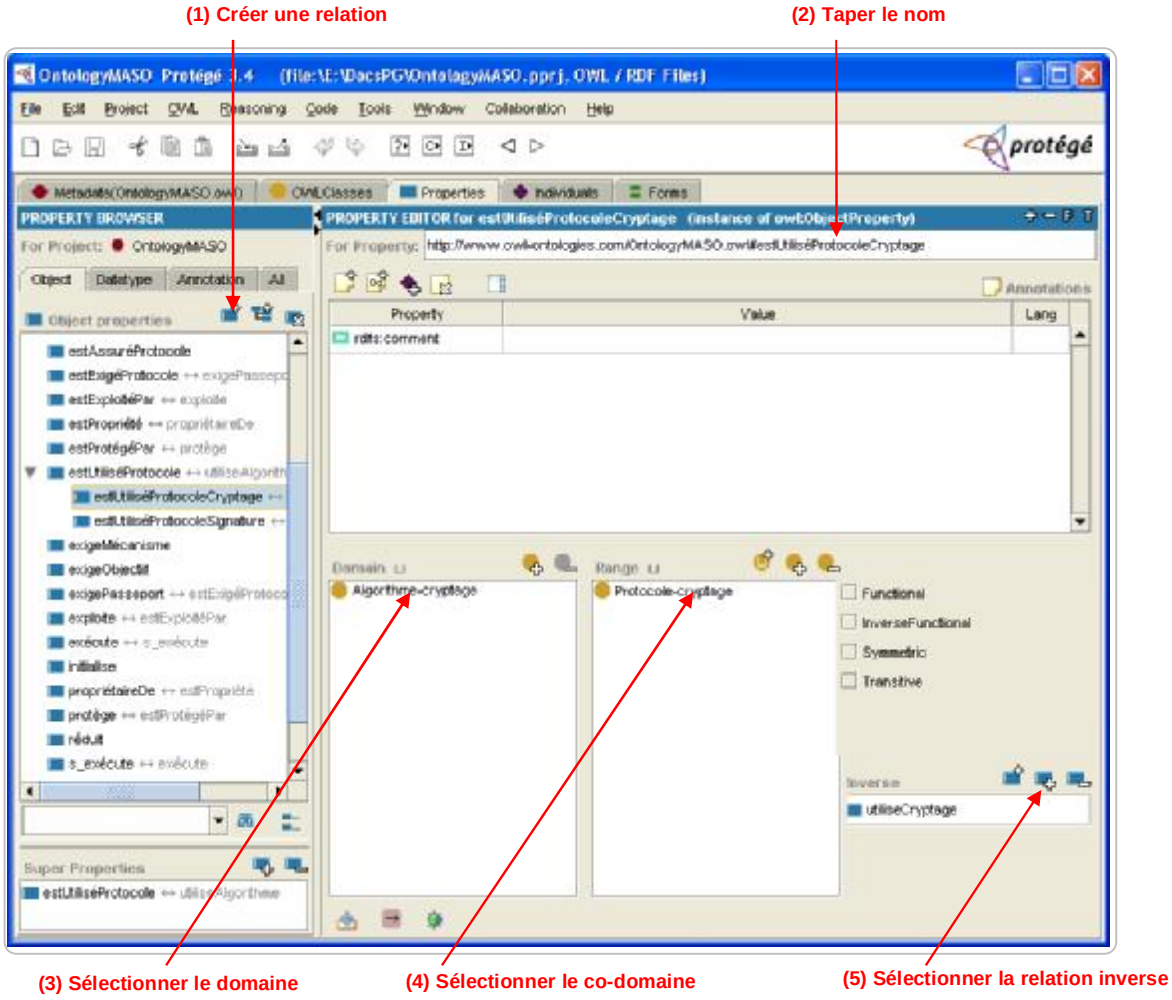

Figure 23 – Création d'une relation

4.4.4 Définitions des restrictions :

PROTEGE OWL nous offre un moyen pour créer des restrictions sur les classes et les propriétés. La Figure 24 montre comment créer une restriction sur une classe.

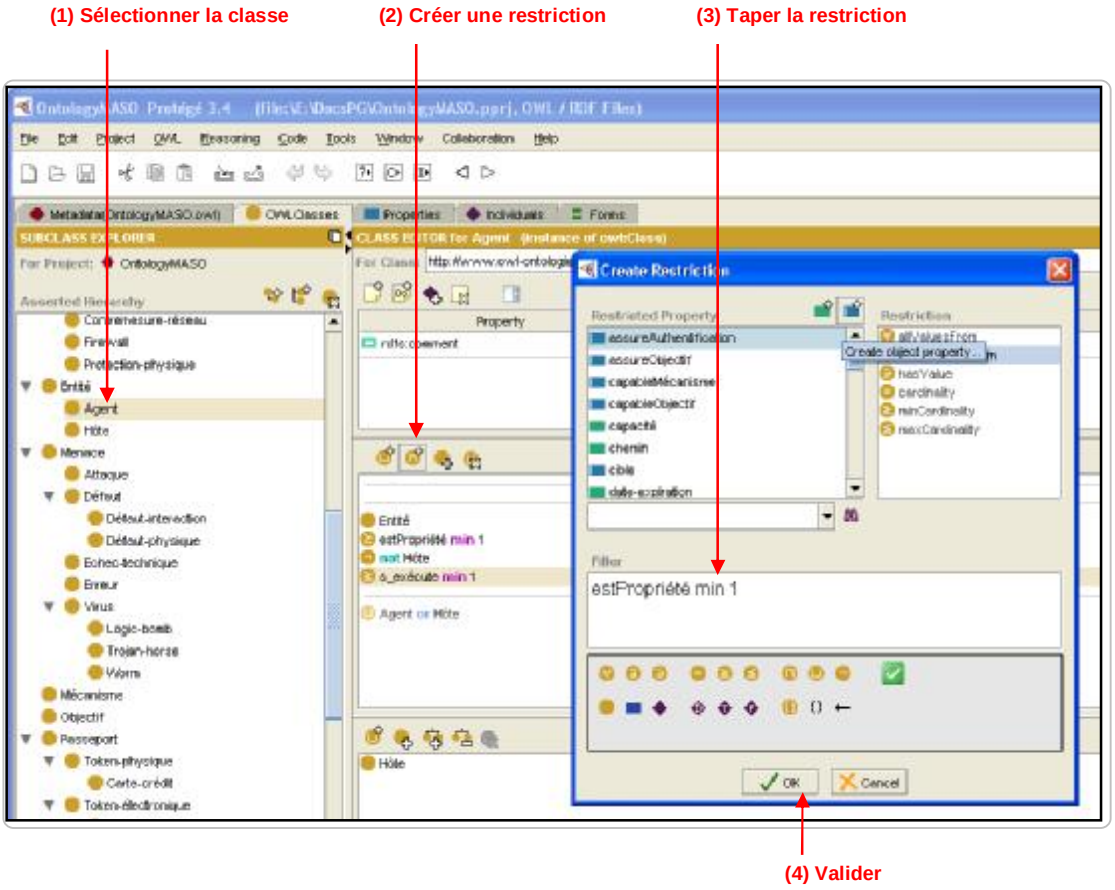

Figure 24 – Création d'une restriction sur une classe

4.4.5 Définition des instances :

La dernière étape consiste à créer les instances des classes dans la hiérarchie. Définir une instance individuelle d'une classe exige

1. choisir une classe
2. créer une instance individuelle de cette classe
3. la renseigner avec les valeurs des attributs.

Par exemple, la Figure 25 montre la création d'un individu de la classe Propriétaire.

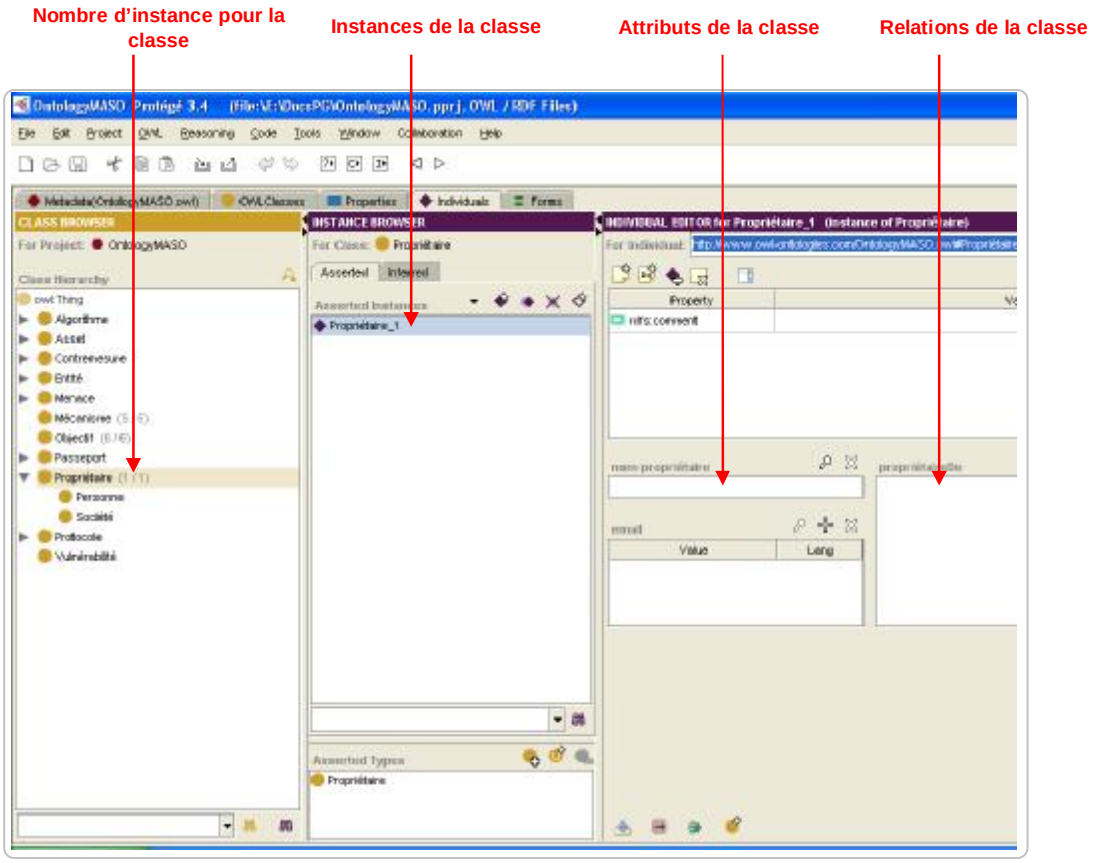

Figure 25 – Création des instances

4.4.6 Génération du code :

L'outil PROTEGE OWL a été conçu pour dégager et libérer le développeur de la complexité du codage, même pour implémenter une petite ontologie, celle–ci va prendre plusieurs lignes du codes et nécessite un grand effort.

Le code peut être visualisé en cliquant sur la commande ***Show RDF/XML Source Code*** dans le menu ***Code***. Le code sera donné à l'annexe de ce mémoire.

4.5 Test de l'ontologie :

Nous avons utilisé le système Racer pour tester l'ontologie MASO, nous distinguons deux types de test: test de consistance et test de satisfiabilité; le premier consiste à enlever l'inconsistance entre les concepts et cela en utilisant le test de subsumption incorporé au

système Racer, par contre le deuxième permet de vérifier pour chaque concept l'existence des instances; un concept C est satisfiable si et seulement si, il existe au moins une interprétation I (instance) pour le concept C.

Racer se présente sous la forme d'un serveur qui peut être accédé par le protocole TCP ou HTTP. Donc, nous devons d'abord configurer la connexion au serveur hébergeant le système Racer. Mais ce que nous souhaitons, est de tester l'ontologie localement. Pour cela nous allons procéder comme suit:

1. Activer le menu ***OWL*** de Protégé.
2. Choisir l'option ***Préférences***…
3. Dans la fenêtre qui s'affiche, vérifier que l'URL est « *http://localhost:8080* », sinon saisir puis valider en cliquant sur le bouton ***'Close'***.

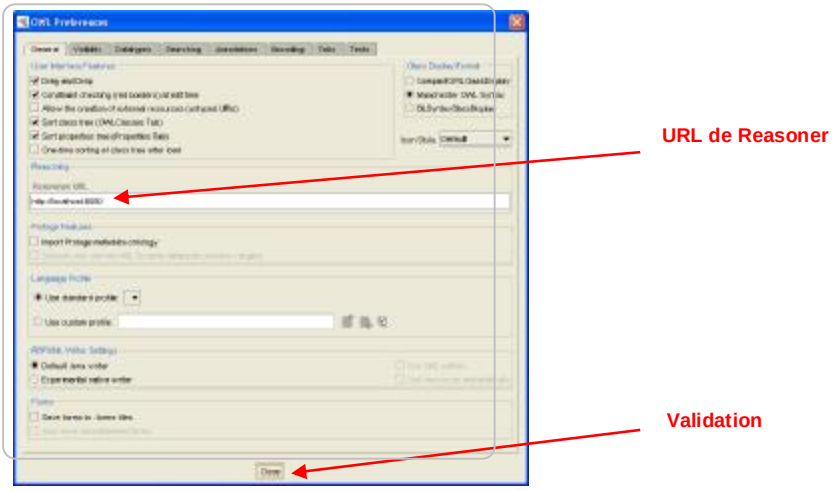

Figure 26 – Vérification de l'URL de Reasoner

4. Télécharger et exécuter Racer localement, le service HTTP va être activé sur le port 8080 de la machine local (localhost)

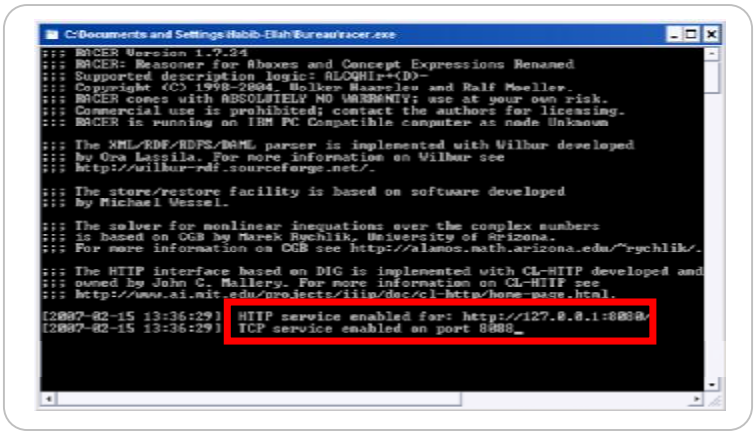

Figure 27 – Lancement de Racer

D'après les tests que nous avons appliqués à l'ontologie MASO, aucune erreur ne s'est produite lors du test.

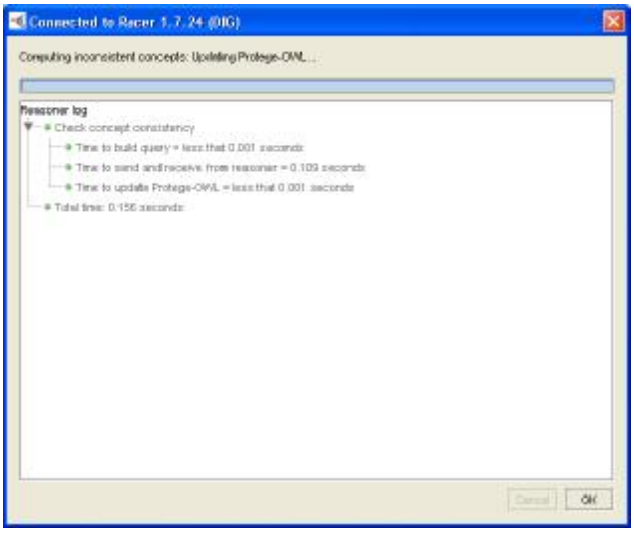

Figure 28 – Test de consistance

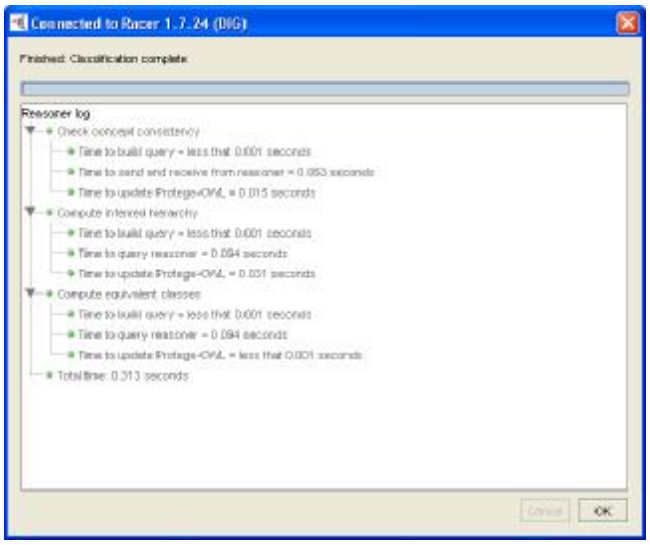

Figure 29 – Test de classification

Après avoir développé notre ontologie MASO, il nous reste à suivre son évolution, c'est-à-dire les nouveaux concepts à ajouter dans la partie terminologique (TBOX). Le résultat de cette étape est une nouvelle ontologie avec une nouvelle hiérarchie de concepts. Pour cela, nous proposons l'utilisation du raisonnement par classification qui est l'un des mécanismes de raisonnements de base pour les logiques de descriptions.

5 CONCLUSION

Dans ce chapitre, nous avons présenté notre contribution, en montrant la place prépondérante que détiennent les ontologies au sein d'un environnement dynamique par quelques scénarios de communication proposés.

Après, nous avons développé une ontologie OWL pour le domaine de la sécurité. Nous avons utilisé un processus de construction d'une ontologie en s'inspirant des différentes phases proposées par la méthode METHONTOLOGY pour la conceptualisation de l'ontologie afin d'atteindre un ensemble de représentations intermédiaires qui facilite sa formalisation ultérieure et cela en adoptant l'approche de la logique de descriptions. Dans ce processus, nous avons utilisé le langage RDF dans la phase de spécification afin d'établir un document formel de spécification des besoins. Nous nous sommes basés sur la formalisation avec les logiques de description, et nous avons choisi le langage OWL pour codifier l'ontologie formelle. Ensuite, nous avons utilisé l'éditeur graphique PROTEGE OWL pour guider l'implémentation et produire un document OWL. Par ailleurs, pour vérifier et raffiner l'ontologie OWL, nous avons utilisé le système RACER. Ce dernier, fournit un support de raisonnement en translatant des expressions OWL à des expressions de la logique de description.

CONCLUSION GÉNÉRALE & PERSPECTIVES

1 CONCLUSION

Les systèmes ouverts utilisant le paradigme d'agent mobile ont besoin, de plus en plus, de souplesse et d'efficacité pour garantir une meilleure communication et réussir l'interopérabilité entre les différentes entités, en particulier en ce qui concerne la sécurité. Désormais, la croissance de complexité et extensibilité au sein des environnements dynamiques empêche l'utilisation d'une politique de sécurité commune et le développement d'une architecture unique préalablement établies. Cela exige une alternative pour pallier au problème d'hétérogénéité en fournissant une sémantique interprétable, claire, et partagée entre les entités communicantes.

2 CONTRIBUTION

Dans ce mémoire, nous avons commencé par la représentation du domaine de sécurité des agents mobiles dans laquelle nous avons présenté la notion d'agent mobile, ses caractéristiques et propriétés. Nous avons découvert les principaux termes et notions correspondants au domaine de sécurité informatique à savoir les attaques, la cryptographie, les protocoles de sécurité, …etc. Nous avons, ensuite, présenté les exigences, problèmes et menaces concernant les agents mobiles et les plates-formes d'agent. Enfin, nous avons présenté les différentes approches utilisées comme contre-mesure. Nous nous sommes concentrés à la présentation des ontologies en définissant leurs composants, types et besoins. Nous avons présenté les trois principaux formalismes de présentation, et détaillé les logiques de description qui constituent notre choix de formalisation. Nous avons découvert quelques méthodologies de construction des ontologies, avec les outils nécessaires à savoir les langages de spécification, les moteurs d'inférences, les éditeurs d'ontologies, …etc. Ensuite, nous avons présenté quelques approches utilisant les ontologies dans le domaine de la sécurité.

S'inscrivant dans le domaine de la sécurité liée aux agents mobiles, et se vouant essentiellement au développement d'une ontologie de domaine de sécurité, ce projet traite le problème de l'hétérogénéité sémantique entre les agents mobiles et leurs plates-formes à travers les propositions suivantes :

2.1 Développement des scénarios de communication entre les agents mobiles et les plates-formes :

Nous avons développé des scénarios de communication en exploitant les ontologies afin d'avoir un contexte de sécurité claire et compréhensible et garantir une meilleure interopérabilité par la possibilité d'adaptation en détectant l'hétérogénéité pouvant exister entre les entités en communication.

2.2 Construction d'une ontologie de domaine dans le cadre de la sécurité des agents mobiles :

Nous avons développé l'ontologie de domaine en suivant les étapes du processus choisi inspiré de METHONTOLOGY, nous avons commencé par la phase de spécification. Ensuite, nous avons organisé et structuré les connaissances acquises en utilisant des représentations intermédiaires semi formels indépendantes de tout langage d'implémentation.

Après, nous avons utilisé le formalisme des logiques de description pour représenter l'ontologie d'application dans un langage formelle. Finalement, nous avons implémenté et testé l'ontologie en utilisant l'éditeur protégé OWL et le raisonneur RACER.

3 EVOLUTION & PERSPECTIVES

La spécification et la mise en œuvre d'une architecture ouverte basée sur les ontologies pour les agents mobiles et leurs plates-formes, même en se limitant aux propriétés de sécurité, est une tâche complexe. De ce fait, nous nous sommes limités, dans ce mémoire, au développement de l'ontologie qui s'inscrit dans le cadre de la sécurité liées aux agents mobiles, avec la proposition, d'une manière générale, de quelques scénarios qui montrent comment l'utiliser.

Nous avons dégagé de ce mémoire quelques perspectives, nous citons celles qui nous ont parrues les plus pertinentes.

3.1 Evolution des scénarios de communication proposés :

Les scénarios de communication que nous avons proposés sont dénués de toutes informations détaillées sur l'acheminement des messages entre les agents mobiles et les plate-formes. L'objectif derrière la proposition des scénarios de communication, qui ne sont pas détaillés, est de montrer l'effet de l'ajout des ontologies au sein d'un SMA.

Des enrichissements et améliorations doivent être apportés à ces scénarios afin de les détailler, et d'avoir une idée plus raffinée sur les interactions garantissant une meilleure interopérabilité sémantique entre les agents mobiles et leurs hôtes.

3.2 Evolution de l'ontologie de domaine :

L'ontologie de domaine de sécurité est construite, il nous reste à suivre son évolution, c'est-à-dire l'ajout de nouveaux concepts dans la partie terminologique (TBOX). Le résultat de cette étape sera une nouvelle ontologie avec une nouvelle hiérarchie de concepts.

Références bibliographiques :

[Acharya97] A. Acharya, M. Ranganathan, Joel Salz, *"Sumatra: A Language for Resourceaware Mobile Programs"*. In J. Vitek and C. Tschudin (Eds.), Mobile Object Systems: Towards the Programmable Internet, Springer-Verlag, Lecture Notes in Computer Science No. 1222, pp. 111-130, April 1997. http://www.cs.umd.edu/~acha/papers/lncs97-1.html

[AmaraH06] N. Amara Hachmi, *"GAMA: une architecture générique pour le développement d'agents mobiles sensibles au contexte et auto-adaptables"*. Thèse pour l'obtention du Doctorat, Laboratoire d'Informatique de Paris Nord, Institut Galiliée, Université Paris XIII, Décembre 2006.

[Anderson72] J. Anderson, *"Computer Security Technology Planning Study"*. U.S. Air Force Electronic Systems Division Technical Report, pp. 51–73, October 1972.

[Baader03a] F. Baader, et W. Nutt, *"Basic description logics"*. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), *"The Description Logic Handbook : Theory, Implementation and Applications"*. Cambridge University Press, pp. 47100. 2003.

[Baader03b] F. Baader, I. Horrocks et U. Sattler, *"Description logics as ontology languages for the semantic web"*. Dans Hutter, D. et Stephan, W. (éditeurs), *"Festschrift in honor of Jörg Siekmann"*. Lecture Notes in Artificial Intelligence. Springer-Verlag. 2003.

[Bachimont03] B. Bachimont, J. Charlet & R. Troncy, *"Ontologies pour le Web Sémantique"*. Action spécifique 32 CNRS / STIC Web sémantique Rapport final. 2003.

[Bachimont04] B. Bachimont, *"Arts et sciences du numérique : Ingénierie des connaissances et critique de la raison computationnelle"*, Mémoire d'Habilitation à Diriger des Recherches, Université de Technologie de Compiègne, 2004.

[Barak01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A Sahai, S. Vadhan and K. Yang, *"On the Impossibility of Obfuscating Programs"*. Advances in Cryptology, Proceedings of Crypto'2001, Lecture Notes in Computer Science, Vol. 2139, pages 1—18, 2001.

[Bechhofer01] S. Bechhofer, I. Horrocks, C. Goble and R. Stevens, *"OILEd: a Reason-able Ontology Editor for the Semantic Web"*, In Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, September 19-21, Vienna. Springer-Verlag LNAI Vol. 2174, pp 396-408. 2001.

[Bella04] P. Bella vista, A. Corradi, C. Frederici, R. Montanari, and D. Tibaldi, *"Security for Mobile Agents: Issues and Challenges"*. Invited Chapter in the Book Handbook of Mobile Computing, I. Mahgoub, M. Ilyas(eds.), CRC Press, 2004.

[Bernard99] G. Bernard. *"Applicabilité et performances des systèmes d'agents mobiles dans les systèmes répartis"*. In Première Conférence Française en Systèmes d'Exploitation (CFSE' 1), Rennes, France, Juin 8-11 1999. 24.

[Blum88] M. Blum and S. Kanan, *"Designing programs to check their work"*. Technical report TR-88-009, International Computer Science Institute, December 1988.

[Borselius02] N. Borselius, *"Mobile Agent Security"*. Electronics Communication Engineering Journal, vol. 14, No 5, IEEE. London, pages 211-218, 2002.

[Brachman84] R. J. Brachrnan, J. Levesque. *"The tractability of subsumption in frame-based description languages"*. In Proceedings of the Fourth National Conference, on Artificial Intelligence, pp. 34-37, Austin, Texas, 1984.

[Borst97] W. N. Borst, *"Construction of engineering ontologies"*. University of Tweenty, Enschede, Centre for Telematica and Information Technology, 1997.

[Carzaniga97] A. Carzaniga, G. P. Picco, and G. Vigna, *"Designing Distributed Applications with Mobile Code Paradigms"*. 19th International Conference on Software Engineering. ACM Press, May 1997.

[Chess94] D. Chess, C. Harrison, and A. Kershenbaum, *"Mobile Agents: Are They a Good Idea?"*. Technical Report RC 19887, IBM Research Division, T.J. Watson Research Center, 1994.

[Collberg97] C. Collberg, C. Thomborson and D. Low, *"A Taxonomy of Obfuscating Transformations"*. In Technical Report 148, Department of Computer Science, University of Auckland, July 1997.

[D'Anna03] L. D'Anna, B. Matt, A. Reisse, T. Van Vleck, S. Schwab and P. LeBlanc, *"Self-Protecting Mobile Agents Obfuscation Report"*. Report #03-015, Network Associates Laboratories Report, 2003.

[Dey00] A. K. Dey and G. D. Abowd, *"Towards a better Understanding of Context and Context-Awareness"*, Workshop on the What, Who, Where, When and How of Context-Awareness, The Hague, The Netherlands, April 2000.

[Farmer96] W. Farmer, J. Guttmann and V. Swarup, *"Security for Mobile Agents: Issues and Requirements"*. In: Proceedings of the National Information Systems Security Conference (NISSC 96), 1996.

[Farquhar96] Farquhar, R. Fikes and J. Rice,*"The Ontolingua Server: A Tool for Collaborative Ontology Construction"*, In Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Alberta, Canada, 44.1-44.19, 1996, http://www.ksl.stanford.edu/software/ontolingua/.

[Fernandez97] M. Fernandez, A. Gomez-Perez et N. Juristo, *"METHONTOLOGY: from ontological art toward ontological engineering"*. Spring symposium series on ontological engineering. AAAI97, USA, 1996.

[Furst02] F. Furst, "L'ingénierie ontologique". Rapport de recherche N°02-07. 2002.

[Gomez-Pérez99] Gomez Pérez A., Benjamins V.R. *"Overview of Knowledge Sharing and Reuse Components : Ontologies and problem-Solving Methods"*. Proceeding of the IJCAI-99 workshop on Ontologies and problem-Solving Methods (KRR5), Stockholm (Suède), pp. 1.1-1.15, 1999.

[Gomez-Pérez02] A. Gómez-Pérez, M. Fernandez-Lopez, O. Corcho, *"OntoWeb: Ontology -based information exchange for knowledge management and electronic commerce"*, Deliverable 1.3: A survey on ontology tools, IST-2000-29243, 31 st May, 2002

[Gong98] L. Gong, *"Secure java class loading. IEEE Internet Computing"*, pages 56 - 61, 1998.

[Gray02] R. S. Gray, G. Cybenko, D. Kotz, R. A. Peterson, and D. Rus,.*"D'agents : Applications and performance of a mobile-agent system"*. Softw., Pract. Exper., 32(6) :543–573, 2002. 24, 41.

[Gruber93] T. Gruber, *"A translation approach to portable ontology specification"*, 1993.

[Gruber95] T.R. Gruber, *"Toward principles for the design of ontologies used for knowledge sharing"*. International Journal of Human Computer Studies. 1995.

[Grüninger95] M. Gruninger and M.S. Fox, *"Methodology for the Design and Evaluation of Ontologies"*. In: Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montreal, 1995.

[Guarino97] N. Guarino, *"Understanding, building, and using ontologies"*. International Journal of Human-Computer Studies, 46: 293-310. 1997.

[Guarino98] N. Guarino, *"Formal Ontology and Information Systems"*. Formal Ontology in Information Systems. IOS Press, 1998.

[Haarslev01] V. Haarslev and R. Möller, *"Racer user's guide and reference manual version 1.6"*. Technical report, University of Hamburg, Computer Science Department, 2001.

[Hacini08] Hacini, *"Sécurité des Systèmes d'Information : Mise en oeuvre de la confiance et de l'adaptabilité pour la protection de l'agent mobile*". Thèse pour l'obtention du Doctorat, Laboratoire LIRE, Université de Mentouri – Constantine, 2008.

[Hayes79] P. J. Hayes, *"The logic of frames"*. In D. Metzing (Ed). Frame Conceptions and Text Understanding. Walter de Gruyter & Co., 1979.

[He98] Qi He, Katia P. Sycara, and Timothy W. Finin. *"Personal security agent: KQML-Based PKI"*. In K. P. Sycara and M. Wooldridge, editors, Proceedings of the 2nd International Conference on Autonomous Agents, pages 377–384, ACM Press, New York, USA, 1998.

[Hemam04]M. Hemam, Z. Boufaida, *"An Ontology Development Process for the Semantic Web"*. EKAW'04, 14th International Conference on Knowledge Engineering and Knowledge Management Whittlebury Hall, Northamptonshire, UK, 5-8 October 2004.

[Hohl97] F. Hohl, *"An approach to solve the problem of malicious hosts"*. Universität Stuttgart, Fakultät Informatik, Fakultätsbericht Nr. 1997/03, 1997. http://www.informatik.unistuttgart.de/cgi-bin/ncstrl_rep_view.pl?/inf/ftp/pub/library/ncstrl.ustuttgart_fi/TR-1997-03/TR-1997-03.bib

[Hohl98] F. Hohl, *"Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts"*. G.Vigna (Ed.), Mobile Agents and Security. Lecture Notes in Computer Science, Vol. 1419, Springer-Verlag, pages 52—59, 1998.

[Hohl00] F. Hohl, *"A framework to Protect Mobile Agents by Using References States"*. In Proceedings of ICDCS 2000, 2000.

[Horrocks98] I. Horrocks, *"The FaCT system"*. Pages 307–312, 1998.

[Horrocks03] I. Horrocks, P. F. Patel-Schneider, et F. van Harmelen, *"From shiq and rdf to owl: The making of a web ontology language"*. J. of Web Semantics 1 (1), 726. 2003.

[Ikeda97] M. Ikeda, K.Seta et al., *"Task Ontology Makes It Easier To Use Authoring Tools"*, Proc. of IJCAI-97, pp.342-347, 1997.

[Ismael99] L. Ismael and D. Hagimont, *"A performance evaluation of mobile agent paradigm"*. In Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA' 99), pages 306–313, November 1999.

[Jansen00] W. Jansen and T. Karygiannis, *"Mobile Agent Security"*. NIST Special Publication 800-19, National Institute of Standard and Technology, 2000.

[Jansen01a] W. Jansen, *"Determining Privileges of Mobile Agents"*. In Proceedings of the Computer Security Applications Conference, December 2001.

[Jansen01b] W. Jansen, *"A Privilege Management Scheme for Mobile Agent Systems"*. First International Workshop on Security of Mobile Multiagent Systems, Autonomous Agents Conference, May 2001.

[Jansen01c] W. Jansen, *"Countermeasures for Mobile Agent Security"*. National Institute of Standards and Technology, Gaithersburg, MD 20899, USA, October 2001.

[Jennings95] N. R. Jennings and M. Wooldridge, *"Intelligent agents: Theory and practice"*. The Knowledge Engineering Review, 10(2): 115–152, 1995.

[Jennings98] N. R. Jennings, M. Wooldridge, and K. Sycara, *"A roadmap of agent research and development"*. Int. Journal of Autonomous Agent and Multi-Agent Systems, vol. 1, n° 1, p. 7-38, 1998.

[Johansen98] D. Johansen, *"Mobile agent applicability"*. Lecture Notes in Computer Science, 1477:80–98, 1998. 24.

[Jun02] L. Jun, L. Xianlang, H. Hong, and Z. Xu, *"Application of mobile agent in wide area network"*. In IEEE, 2002. 24.

[Kagal02] L. Kagal, *"Rei : A policy language for the me-centric project"*. Technical report, HP Labs. 2002.

[Karnik98] N. Karnik, *"Security in Mobile Agents Systems"*. PhD thesis, Department of Computer Sciences and Engineering, University of Minnesota, Minneapolis, USA, 1998.

[Kotz99] D. Kotz and R. S. Gray, *"Mobile agents and the future of the internet"*. Operating Systems Review, 33(3):7–13, July 1999. 24.

[Lange99] D.B. Lange and M. Oshima, *"Seven good reason for mobile agents"*. Communication of the ACM, 42, 88-89, 1999.

[Lee02] T.B. Lee et al., "The semantic Web". in Scientific American, May 2002.

[Loureiro01a] S. Loureiro, R. Molva and Y. Roudier, *"Mobile Code Security"*. Institut Eurecom, 2001.

[Loureiro01b] S. Loureiro, *"Mobile code protection"*. Thèse de doctorat, Institut Eurocom, January 2001.

[Maes94] M. Maes, *"Modeling adaptative autonomous agent"*. Artificial life journal, 1(1 et 2), pages 135-162, 1994.

[Maes99] P. Maes, R. H. Guttman, and A.G. Moukas, *"Agents that buy and sell"*. Communication of the ACM, 42(3):81–91, March 1999.

[Mana02] A. Mana and E. Pimentel, *"An efficient software protection scheme"*. University of Malaga, Spain, 2002.

[Minsky75] M. Minsky, *"A framework for representing knowledge"*. The Psychology of Computer Vision. McGraw-Hill. 1975.

[Minsky81] M. Minsky, *"A framework for representing knowledge"*. Dans Haugeland, J. (éditeur), Mind Design. The MIT Press, pp. 95128. 1981.

[Nardi03] D. Nardi et R. J. Brachman. "*An introduction to description logics*". Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), *"The Description Logic Handbook : Theory, Implementation and Applications"*. Cambridge University Press, pp. 544. 2003.

[Neches91] R. Neches, R.E. Fikes, T. Finin, T. Gruber, T. Senator, W.R. Swartout, *"Enabling technology for knowledge sharing"*, AI Magazine, 1991.

[Necula98] G. C. Necula, P. Lee, *"Untrusted Agents using Proof-Carring Code"*. Lecture Notes in Computer Science, Vol. 1419, Springer-Verlag, 1998.

[Noy00] N. Noy, R. W. Fergerson and M. A.Musen. *"The knowledge model of Protégé2000: combining interoperability and flexibility"*. In Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW'00), 2000.

[Noy01] Natalya F. Noy and Deborah L. McGuinness, *"Ontology Development 101: A Guide to Creating Your First Ontology"*. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

[Oberle04] D. Oberle, R. Volz, B. Motik et S. Staab, *"An extensible ontology software environment"*. In S. Staab et R. Studer (Eds.), Handbook on Ontologies (pp. 299-320): Springer Verlag. 2004.

[Quillian68] M. Quillian, *"Semantic memory"*. Semantic Information Processing. MIT Press, Cam. 1968.

[Reiser00] H. Reiser, *"Security Requirements for Management Systems using Mobile Agents"*. Proceeding of the Fifth IEEE Symposium on Computers and Communications: ISCC 2000, Antibes, France, pages 160-165, 2000.

[Riordan98] J. Riordan, B. Schneier, *"Environment key Generation towards Clueless Agents"*. Lecture Notes in Computer Science, Vol. 1419, pages 15—24, 1998.

[Roth99] V. Roth, *"Mutual Protection of Cooperating Agents"*. Secure Internet Programming: Security Issues for Mobile and Distributed Objects. J. Vitek and C. Jensen (Eds.), Springer Verlag, 1999.

[Rouvrais02] Siegfried Rouvrais. *"Utilisation d'Agents Mobiles pour la Construction de Services Distribués"*. Thèse, Université de Rennes 1, 2002. 14, 25, 35.

[Sahai98] A. Sahai and C. Morin, *"Mobile agents for enabling mobile user aware applications"*. In Katia P. Sycara and Michael Wooldridge, editors, Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98), pages 205–211, New York, May 9–13, 1998. ACM Press. 24, 47.

[Sander98a] Sander, T., Tschudin, C., *"Toward Mobile Cryptography. IEEE Symposium Security and Privacy"*. IEEE Computer Soc. Press, Los Alamitos, California, pages 215-224, 1998.

[Sander98b] T. Sander, C. Tschudin, *"Protecting Mobile Agent against Malicious Hosts"*. G.Vigna (Ed.), Mobile Agents and Security, Lecture Notes in Computer Science, Vol. 1419, ©Springer-Verlag Berlin Heidelberg, Berlin, pages 44 - 60, 1998.

[Sandhu98] R. S. Sandhu, *"Role-based access control"*. Advances in Computers, 46. 1998.

[Sattler03] U. Sattler, D. Calvanese, et R. Molitor, *"Relationships with other formalisms"*. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), *"The Description Logic Handbook: Theory, Implementation and Applications"*. Cambridge University Press, pp. 142183. 2003.

[Sayeb02] M. Sayeb, M. Hamza, A. Soliman, *"Protecting Mobile Agents against Malicious Host Attacks Using Treat Diagnostic AND/OR Tree"*. Arab Academy for Science, Technology & Maritime Transport Computer Engineering Department, Alexandria, Egypt, 2002.

[Schmidt-Schauss91] M. Schmidt-Schaub, et G. Smolka, *"Attributive concept descriptions with complements"*. Artificial Intelligence 48 (1), 126. 1991.

[Sirin06] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, *"Pellet : A practical owl-dl reasoned"*. Submitted for publication to Journal of Web Semantics, 2006.

[Sowa84] J. Sowa, *"Conceptual Structures: information processing in mind and machine"*. Addison-Wesley, 1984.

[Staab00] S. Staab, A. Maedche, *"Axioms are objects too: Ontology engineering beyond the modeling of concepts and relations"*, Research report 399, Institute AIFB, Karlsruhe, 2000.

[Sure02] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer and Wenke, D., *"OntoEdit: Collaborative Ontology Engineering for the Semantic Web"*. In Proceedings of the International Semantic Web Conference 2002 (ISWC 2002), Sardinia, Italia, June 2002.

[Swartout97] B. Swartout, P. Ramesh, K. Knight and T. Russ, *"Toward Distributed Use of Large-Scale Ontologies"*, In Symposium on Ontological Engineering of AAAI, Stanford, California, March, 1997.

[Swarup97] V. Swarup, *"Trust Appraisal and Secure Routing of Mobile Agents"*. DARPA Workshop on Foundations for Secure Mobile Code, Position Paper. Monterey, CA, USA, March 1997.

[Tan01a] Tan, H. K., Moreau, L., *"Trust Relationships in a Mobile Agent System"*. In G. P. Picco, editor, Fifth IEEE International Conference on Mobile Agents, Lecture Notes in Computer Science, vol. 2240, Springer-Verlag, Atlanta, Georgia, 2001.

[Tan02b] H. K. Tan, L. Moreau, D. Cruickshank, and D. De Roure, *"Certificates for Mobile Code Security"*. In Proceedings of The 17th ACM Symposium on Applied Computing (SAC'2002) - Track on Agents, Interactions, Mobility and Systems, page 76. 2002.

[Tennenhouse96] D. Tennenhouse and D. Wetheral, *"Towards an active network architecture"*. In ACM Computer Communications Review, volume 26, pages 5{18, April 1996.

[Tonti03] G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok, *"Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder"*. In Proceedings of the International Semantic Web Conference 2003, pages 419–437, Sanibel Island, Florida. 2003.

[Troncy04] Raphaël Troncy, *"Formalisation des connaissances documentaires et des connaissances conceptuelles à l'aide d'ontologies : application à la description de documents audiovisuels"*. Thèse pour l'obtention du Doctorat de l'université Joseph Fourier – Grenoble, 2004.

[Tsarkov04] D. Tsarkov and I. Horrocks, *"Efficient reasoning with range and domain constraints"*. In Proc. of the 2004 Description Logic Workshop (DL 2004), pages 41–50, 2004.

[Uschold96] M. Uschold and M. Grüninger, *"ONTOLOGIES: Principles, Methods and Applications"*. Knowledge Engineering Review, 1996.

[Uschold02] M. Uschold and M.Gruninger, *"Creating semantically integrated communities on the World Wide Web"*. Honolulu: Semantic Web Workshop, 2002

[Uszok03] A. Uszok, J. M. Bradshaw, R. Jeffers, N. Suri, P. J. Hayes, M. R. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott, *"Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement"*. In Proceedings of POLICY 2003, Como, Italy. 2003.

[Vigna97] G. Vigna, *"Protecting mobile agents through tracing"*. In Proceedings of the Third ECOOP Workshop on Operating System support for Mobile Object Systems, Finland, pages 137-153, June 1997.

[Vigna98] G. Vigna, edition, *"Mobile Code Security"*. Lecture Notes in Computer Science, Vol.1419, Springer-Verlag Berlin Heidelberg, Berlin, 1998.

[Vigna04] G. Vigna, *"Mobile Agents: Ten Reasons for failure"*. In 5th IEEE International Conference on Mobile Data Management (MDM 2004), pages 298-299. IEEE Computer Society, pages 19-22, January 2004.

[Wahbe93] R. Wahbe, S. Lucco, T. E. Anderson and S. L. Graham, *"Efficient software-based fault isolation"*. In Proceedings of the 14th ACM Symposium on Operating Systems Principles, pages 203-216, December 1993.

[Wilhelm97] U. G. Wilhelm, *"Cryptographically Protected Objects"*. A french version appeared in the Proceedings of RenPar'9. http://lsewww.ep.ch/wilhelm/Papers/CryPO.ps.gz

[Wilhelm98] U. G. Wilhelm, S. Staamann and L. Buttyan, *"On the Problem of Trust in Mobile Agent Systems"*. IEEE Symposium on Network and Distributed System Security, San Diego, California, 1998.

[Zou04] Y. Zou, T. Finin, et H. Chen, *"F-owl : an inference engine for semantic web"*. Dans Third NASA-Goddard/IEEE Workshop on Formal Approaches to Agent-Based Systems. Greenbelt, Maryland. 2004.

Sites Web :

[ht1] http://fr.wikipedia.org/wiki/Agent(informatique).htm, Novembre 2007.

[ht2] http://www.gelgabon.net/securite-sauvegarde/introduction-a-la-securite-informatique, Janvier 2009.

[ht3] http://fr.wikipedia.org/wiki/Politique_de_s%C3%A9curit%C3%A9_informatique, Janvier 2009.

[ht4] http://protege.stanford.edu, Juin 2009

[W3C04a] Recommendation W3C RDF, 2004. http://www.w3.org/TR/rdf-syntax-grammar/.

[W3C04b] Recommendation W3C RDF Schema, 2004. http://www.w3.org/TR/rdf-schema/.

[W3C04c] Recommendation W3C OWL, 2004. http://www.w3.org/TR/owl-ref/.

[W3C04d] Soumission W3C RDQL, 2004. http://www.w3.org/Submission/RDQL/.

[W3C06] Candidate recommandation W3C SPARQL, 2006. http://www.w3.org/TR/rdf-sparql-query/.

Glossaire :

AES	Advanced Encryption Standard
B2B	Business To Business
CPU	Central Processing Unit
C-SET	Chip-Secure Electronic Transaction
DAML-OIL	DARPA Agent Markup Language OIL
DES	Data Encryption Standard
DoS	Denial of Service
DSA	Digital Signature Algorithm
HTML	Hyper Text Markup Language
IDEA	International Data Encryption Algorithm
IGC	Infrastructure de Gestion de Clés
KAoS	Knowledge Acquisition in autOmated Specification
MAS	Multi Agent System
nRQL	new Racer query Language
OIL	Ontology Inference Layer
OML	Ontology Markup Language
OWL	Ontology Web Language
PCC	Proof Carrying Code
PDA	Personal Digital Assistant
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
RACER	Renamed Abox and Concept Expression Reasoner
RBAC	Role-Based Access Control
RC4	Rivest Cipher 4
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RDQL	RDF Data Query Language
RSA	Rivest Shamir Adleman
S/MIME	Secure/ Multipurpose Internet Mail Extensions
SET	Secure Electronic Transaction
SHOE	Simple HTML Ontology Extensions
SMA	Système Multi Agent
SPARQL	SPARQL Protocol And RDF Query Language
SQL	Structured Query Language
SSL	Secure Sockets Layer
SWRL	Semantic Web Rule Language
TPE	Trusted Processing Environment
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XOL	Ontology Exchange Language

Annexe

L'ontologie MASO en OWL:

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
   <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
   <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
]>

<rdf:RDF xmlns="http://www.owl-ontologies.com/OntologyMASO.owl#"
    xml:base="http://www.owl-ontologies.com/OntologyMASO.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <owl:Ontology rdf:about=""/>
   <Cryptage-symétrique rdf:ID="_3DES"/>
   <owl:Class rdf:ID="Adresse">
      <rdfs:subClassOf rdf:resource="#Token-électronique"/>
   </owl:Class>
   <owl:Class rdf:ID="Adresse-IP">
      <rdfs:subClassOf rdf:resource="#Adresse"/>
   </owl:Class>
   <Cryptage-symétrique rdf:ID="AES"/>
   <owl:Class rdf:ID="Agent">
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#s_exécute"/>
            <owl:minCardinality rdf:datatype="&xsd;int">1</owl:minCardinality>
         </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#estPropriété"/>
            <owl:minCardinality rdf:datatype="&xsd;int">1</owl:minCardinality>
         </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <owl:Class>
            <owl:complementOf rdf:resource="#Hôte"/>
         </owl:Class>
      </rdfs:subClassOf>
      <rdfs:subClassOf rdf:resource="#Entité"/>
      <owl:disjointWith rdf:resource="#Hôte"/>
   </owl:Class>
   <owl:Class rdf:ID="Algorithme">
      <rdfs:label rdf:datatype="&xsd;string"></rdfs:label>
   </owl:Class>
   <owl:Class rdf:ID="Algorithme-checksum">
      <rdfs:subClassOf rdf:resource="#Algorithme"/>
   </owl:Class>
   <owl:Class rdf:ID="Algorithme-cryptage">
      <rdfs:subClassOf rdf:resource="#Algorithme"/>
   </owl:Class>
   <owl:Class rdf:ID="Algorithme-gestion-clés">
      <rdfs:subClassOf rdf:resource="#Algorithme"/>
   </owl:Class>
   <owl:Class rdf:ID="Algorithme-signature">
      <rdfs:subClassOf rdf:resource="#Algorithme"/>
```

```
</owl:Class>
<owl:Class rdf:ID="Antivirus">
   <rdfs:subClassOf rdf:resource="#Contremesure"/>
</owl:Class>
<owl:Class rdf:ID="Asset"/>
<owl:ObjectProperty rdf:ID="assureAuthentification">
   <rdfs:domain rdf:resource="#Protocole-authentification"/>
   <rdfs:range rdf:resource="#Objectif"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="assureObjectif">
   <rdfs:domain>
      <owl:Class>
         <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Mécanisme"/>
            <owl:Class rdf:about="#Protocole"/>
         </owl:unionOf>
      </owl:Class>
   </rdfs:domain>
   <rdfs:range rdf:resource="#Objectif"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Attaque">
   <rdfs:subClassOf rdf:resource="#Menace"/>
</owl:Class>
<Mécanisme rdf:ID="Authentification">
   <assureObjectif rdf:resource="#Confidentialité"/>
</Mécanisme>
<Mécanisme rdf:ID="Autorisation"/>
<Antivirus rdf:ID="AVG"/>
<Cryptage-symétrique rdf:ID="BlowFish"/>
<owl:ObjectProperty rdf:ID="capableMécanisme">
   <rdfs:domain rdf:resource="#Entité"/>
   <rdfs:range rdf:resource="#Mécanisme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="capableObjectif">
   <rdfs:domain rdf:resource="#Entité"/>
   <rdfs:range rdf:resource="#Objectif"/>
</owl:ObjectProperty>
<owl:FunctionalProperty rdf:ID="capacité">
   <rdf:type rdf:resource="&owl;DatatypeProperty"/>
   <rdfs:domain>
      <owl:Class>
         <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Mémoire"/>
            <owl:Class rdf:about="#Storage"/>
         </owl:unionOf>
      </owl:Class>
   </rdfs:domain>
   <rdfs:range rdf:resource="&xsd;int"/>
</owl:FunctionalProperty>
<owl:Class rdf:ID="Carte-crédit">
   <rdfs:subClassOf rdf:resource="#Token-physique"/>
</owl:Class>
<Cryptage-symétrique rdf:ID="CAST"/>
<owl:Class rdf:ID="Certificat">
   <rdfs:subClassOf rdf:resource="#Token-électronique"/>
</owl:Class>
<owl:FunctionalProperty rdf:ID="chemin">
   <rdf:type rdf:resource="&owl;DatatypeProperty"/>
   <rdfs:domain rdf:resource="#Cookie"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:FunctionalProperty>
<owl:ObjectProperty rdf:ID="cible">
   <rdfs:domain rdf:resource="#Menace"/>
   <rdfs:range rdf:resource="#Asset"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Clé-de-cryptage">
   <rdfs:subClassOf rdf:resource="#Token-électronique"/>
</owl:Class>
<owl:Class rdf:ID="Clé-privée">
   <rdfs:subClassOf rdf:resource="#Clé-de-cryptage"/>
</owl:Class>
<owl:Class rdf:ID="Clé-publique">
   <rdfs:subClassOf rdf:resource="#Clé-de-cryptage"/>
```

```
</owl:Class>
<owl:Class rdf:ID="Clé-secrète">
   <rdfs:subClassOf rdf:resource="#Clé-de-cryptage"/>
</owl:Class>
<owl:Class rdf:ID="Composant">
   <rdfs:subClassOf rdf:resource="#Software"/>
</owl:Class>
<Objectif rdf:ID="Confiance">
   <estAssuréMécanisme rdf:resource="#Identification"/>
</Objectif>
<Objectif rdf:ID="Confidentialité"/>
<owl:Class rdf:ID="Contremesure"/>
<owl:Class rdf:ID="Contremesure-réseau">
   <rdfs:subClassOf rdf:resource="#Contremesure"/>
</owl:Class>
<Mécanisme rdf:ID="Contrôle-daccès">
   <assureObjectif rdf:resource="#Disponibilité"/>
</Mécanisme>
<owl:Class rdf:ID="Cookie">
   <rdfs:subClassOf rdf:resource="#Token-électronique"/>
</owl:Class>
<owl:Class rdf:ID="Cryptage-asymétrique">
   <rdfs:subClassOf rdf:resource="#Algorithme-cryptage"/>
</owl:Class>
<owl:Class rdf:ID="Cryptage-symétrique">
   <rdfs:subClassOf rdf:resource="#Algorithme-cryptage"/>
</owl:Class>
<owl:Class rdf:ID="Data">
   <rdfs:subClassOf rdf:resource="#Asset"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="date-expiration">
   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
   <rdfs:domain rdf:resource="#Carte-crédit"/>
   <rdfs:range rdf:resource="&xsd;date"/>
</owl:DatatypeProperty>
<Cryptage-symétrique rdf:ID="DES"/>
<owl:DatatypeProperty rdf:ID="description-logiciel">
   <rdfs:domain rdf:resource="#Software"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<Objectif rdf:ID="Disponibilité"/>
<owl:ObjectProperty rdf:ID="dispose">
   <rdfs:domain rdf:resource="#Hôte"/>
   <rdfs:range rdf:resource="#Asset"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Domaine">
   <rdfs:subClassOf rdf:resource="#Adresse"/>
</owl:Class>
<owl:Class rdf:ID="Défaut">
   <rdfs:subClassOf rdf:resource="#Menace"/>
</owl:Class>
<owl:Class rdf:ID="Défaut-interaction">
   <rdfs:subClassOf rdf:resource="#Défaut"/>
</owl:Class>
<owl:Class rdf:ID="Défaut-physique">
   <rdfs:subClassOf rdf:resource="#Défaut"/>
</owl:Class>
<Attaque rdf:ID="Déni_de_service"/>
<owl:Class rdf:ID="Echec-technique">
   <rdfs:subClassOf rdf:resource="#Menace"/>
</owl:Class>
<Cryptage-asymétrique rdf:ID="ElGamal"/>
<Cryptage-asymétrique rdf:ID="Elliptic_curve"/>
<owl:DatatypeProperty rdf:ID="email">
   <rdfs:domain rdf:resource="#Propriétaire"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="enInteractionAvec">
   <rdfs:domain rdf:resource="#Entité"/>
   <rdfs:range rdf:resource="#Entité"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Entité">
   <rdfs:subClassOf>
```

```
      <owl:Class>
         <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Agent"/>
            <owl:Class rdf:about="#Hôte"/>
         </owl:unionOf>
      </owl:Class>
   </rdfs:subClassOf>
   <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<owl:Class rdf:ID="Erreur">
   <rdfs:subClassOf rdf:resource="#Menace"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="estAssuréMécanisme">
   <rdfs:domain rdf:resource="#Objectif"/>
   <rdfs:range rdf:resource="#Mécanisme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="estAssuréProtocole">
   <rdfs:domain rdf:resource="#Objectif"/>
   <rdfs:range rdf:resource="#Protocole"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="estExigéProtocole">
   <rdfs:domain rdf:resource="#Passeport"/>
   <owl:inverseOf rdf:resource="#exigePasseport"/>
   <rdfs:range rdf:resource="#Protocole"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="estExploitéPar">
   <rdfs:domain rdf:resource="#Vulnérabilité"/>
   <owl:inverseOf rdf:resource="#exploite"/>
   <rdfs:range rdf:resource="#Menace"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="estPropriété">
   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
   <rdfs:domain rdf:resource="#Agent"/>
   <owl:inverseOf rdf:resource="#propriétaireDe"/>
   <rdfs:range rdf:resource="#Propriétaire"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="estProtégéPar">
   <rdfs:domain rdf:resource="#Asset"/>
   <owl:inverseOf rdf:resource="#protège"/>
   <rdfs:range rdf:resource="#Contremesure"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="estUtiliséProtocole">
   <rdf:type rdf:resource="&owl;SymmetricProperty"/>
   <rdfs:domain rdf:resource="#Algorithme"/>
   <owl:inverseOf rdf:resource="#estUtiliséProtocole"/>
   <rdfs:range rdf:resource="#Protocole"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="estUtiliséProtocoleCryptage">
   <rdfs:domain rdf:resource="#Algorithme-cryptage"/>
   <owl:inverseOf rdf:resource="#utiliseCryptage"/>
   <rdfs:range rdf:resource="#Protocole-cryptage"/>
   <rdfs:subPropertyOf rdf:resource="#estUtiliséProtocole"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="estUtiliséProtocoleSignature">
   <rdfs:domain rdf:resource="#Algorithme-signature"/>
   <owl:inverseOf rdf:resource="#utiliseSignature"/>
   <rdfs:range rdf:resource="#Protocole-signature"/>
   <rdfs:subPropertyOf rdf:resource="#estUtiliséProtocole"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="exigeMécanisme">
   <rdfs:domain rdf:resource="#Entité"/>
   <rdfs:range rdf:resource="#Mécanisme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="exigeObjectif">
   <rdfs:domain rdf:resource="#Entité"/>
   <rdfs:range rdf:resource="#Objectif"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="exigePasseport">
   <rdfs:domain rdf:resource="#Protocole"/>
   <owl:inverseOf rdf:resource="#estExigéProtocole"/>
   <rdfs:range rdf:resource="#Passeport"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="exploite">
```

```
    <rdfs:domain rdf:resource="#Menace"/>
    <owl:inverseOf rdf:resource="#estExploitéPar"/>
    <rdfs:range rdf:resource="#Vulnérabilité"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="exécute">
    <rdfs:domain rdf:resource="#Hôte"/>
    <owl:inverseOf rdf:resource="#s_exécute"/>
    <rdfs:range rdf:resource="#Agent"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="fax">
    <rdfs:domain rdf:resource="#Propriétaire"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="Firewall">
    <rdfs:subClassOf rdf:resource="#Contremesure"/>
</owl:Class>
<Mécanisme rdf:ID="Gestion-clés"/>
<owl:Class rdf:ID="Hardware">
    <rdfs:subClassOf rdf:resource="#Asset"/>
</owl:Class>
<owl:Class rdf:ID="Hash">
    <rdfs:subClassOf rdf:resource="#Algorithme-signature"/>
</owl:Class>
<MAC rdf:ID="HMAC"/>
<owl:Class rdf:ID="Hôte">
    <rdfs:subClassOf>
        <owl:Class>
            <owl:complementOf rdf:resource="#Agent"/>
        </owl:Class>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Entité"/>
    <owl:disjointWith rdf:resource="#Agent"/>
</owl:Class>
<Cryptage-symétrique rdf:ID="IDEA"/>
<Mécanisme rdf:ID="Identification"/>
<Objectif rdf:ID="Imputabilité"/>
<owl:ObjectProperty rdf:ID="initialise">
    <rdfs:domain rdf:resource="#Entité"/>
    <rdfs:range rdf:resource="#Menace"/>
</owl:ObjectProperty>
<Attaque rdf:ID="Intrusion"/>
<Objectif rdf:ID="Intégrité"/>
<owl:FunctionalProperty rdf:ID="issuer">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:domain rdf:resource="#X.509"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:FunctionalProperty>
<Algorithme-gestion-clés rdf:ID="KEA"/>
<owl:FunctionalProperty rdf:ID="longueur-clé">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Clé-de-cryptage"/>
    <rdfs:range rdf:resource="&xsd;int"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="longueur-minimum">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Mot-de-passe"/>
    <rdfs:range rdf:resource="&xsd;int"/>
</owl:FunctionalProperty>
<owl:Class rdf:ID="MAC">
    <rdfs:subClassOf rdf:resource="#Algorithme-signature"/>
</owl:Class>
<owl:FunctionalProperty rdf:ID="marque">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Microprocesseur"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:FunctionalProperty>
<Hash rdf:ID="MD4"/>
<Hash rdf:ID="MD5"/>
<owl:Class rdf:ID="Menace"/>
<owl:Class rdf:ID="Microprocesseur">
    <rdfs:subClassOf rdf:resource="#Hardware"/>
</owl:Class>
<owl:FunctionalProperty rdf:ID="modèle">
```

```
   <rdf:type rdf:resource="&owl;DatatypeProperty"/>
   <rdfs:domain rdf:resource="#Microprocesseur"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:FunctionalProperty>
<owl:Class rdf:ID="Mot-de-passe">
   <rdfs:subClassOf rdf:resource="#Token-électronique"/>
</owl:Class>
<owl:Class rdf:ID="Mécanisme">
   <rdfs:subClassOf>
      <owl:Class>
         <owl:unionOf rdf:parseType="Collection">
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Authentification"/>
               </owl:oneOf>
            </owl:Class>
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Autorisation"/>
               </owl:oneOf>
            </owl:Class>
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Contrôle-daccès"/>
               </owl:oneOf>
            </owl:Class>
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Gestion-clés"/>
               </owl:oneOf>
            </owl:Class>
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Identification"/>
               </owl:oneOf>
            </owl:Class>
         </owl:unionOf>
      </owl:Class>
   </rdfs:subClassOf>
   <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<owl:Class rdf:ID="Mémoire">
   <rdfs:subClassOf rdf:resource="#Hardware"/>
</owl:Class>
<owl:FunctionalProperty rdf:ID="nom">
   <rdf:type rdf:resource="&owl;DatatypeProperty"/>
   <rdfs:domain>
      <owl:Class>
         <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Antivirus"/>
            <owl:Class rdf:about="#Cookie"/>
            <owl:Class rdf:about="#Propriétaire"/>
            <owl:Class rdf:about="#Software"/>
            <owl:Class rdf:about="#Virus"/>
         </owl:unionOf>
      </owl:Class>
   </rdfs:domain>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:FunctionalProperty>
<Objectif rdf:ID="Non-répudiation"/>
<Antivirus rdf:ID="Norton"/>
<owl:FunctionalProperty rdf:ID="notAfter">
   <rdf:type rdf:resource="&owl;DatatypeProperty"/>
   <rdfs:domain rdf:resource="#X.509"/>
   <rdfs:range rdf:resource="&xsd;date"/>
</owl:FunctionalProperty>
<owl:DatatypeProperty rdf:ID="notBefore">
   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
   <rdfs:domain rdf:resource="#X.509"/>
   <rdfs:range rdf:resource="&xsd;date"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="numéro-de-série">
   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
```

```
   <rdfs:domain rdf:resource="#X.509"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="Objectif">
   <rdfs:subClassOf>
      <owl:Class>
         <owl:unionOf rdf:parseType="Collection">
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Confiance"/>
               </owl:oneOf>
            </owl:Class>
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Confidentialité"/>
               </owl:oneOf>
            </owl:Class>
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Disponibilité"/>
               </owl:oneOf>
            </owl:Class>
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Imputabilité"/>
               </owl:oneOf>
            </owl:Class>
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Intégrité"/>
               </owl:oneOf>
            </owl:Class>
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Non-répudiation"/>
               </owl:oneOf>
            </owl:Class>
         </owl:unionOf>
      </owl:Class>
   </rdfs:subClassOf>
   <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<owl:Class rdf:ID="Passeport"/>
<owl:Class rdf:ID="Personne">
   <rdfs:subClassOf rdf:resource="#Propriétaire"/>
   <owl:disjointWith rdf:resource="#Société"/>
</owl:Class>
<owl:Class rdf:ID="Propriétaire"/>
<owl:ObjectProperty rdf:ID="propriétaireDe">
   <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
   <rdfs:domain rdf:resource="#Propriétaire"/>
   <owl:inverseOf rdf:resource="#estPropriété"/>
   <rdfs:range rdf:resource="#Agent"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Protection-physique">
   <rdfs:subClassOf rdf:resource="#Contremesure"/>
</owl:Class>
<owl:Class rdf:ID="Protocole"/>
<owl:Class rdf:ID="Protocole-authentification">
   <rdfs:subClassOf rdf:resource="#Protocole"/>
</owl:Class>
<owl:Class rdf:ID="Protocole-cryptage">
   <rdfs:subClassOf rdf:resource="#Protocole"/>
</owl:Class>
<owl:Class rdf:ID="Protocole-gestion-clés">
   <rdfs:subClassOf rdf:resource="#Protocole"/>
</owl:Class>
<owl:Class rdf:ID="Protocole-gestion-réseau">
   <rdfs:subClassOf rdf:resource="#Protocole"/>
</owl:Class>
<owl:Class rdf:ID="Protocole-signature">
   <rdfs:subClassOf rdf:resource="#Protocole"/>
</owl:Class>
```

```
<owl:ObjectProperty rdf:ID="protège">
   <rdfs:domain rdf:resource="#Contremesure"/>
   <owl:inverseOf rdf:resource="#estProtégéPar"/>
   <rdfs:range rdf:resource="#Asset"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="prénom">
   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
   <rdfs:domain rdf:resource="#Personne"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="RBAC">
   <rdfs:subClassOf rdf:resource="#Certificat"/>
</owl:Class>
<Rivest rdf:ID="RC2"/>
<Rivest rdf:ID="RC4"/>
<Rivest rdf:ID="RC5"/>
<Rivest rdf:ID="RC6"/>
<owl:Class rdf:ID="Rivest">
   <rdfs:subClassOf rdf:resource="#Cryptage-asymétrique"/>
</owl:Class>
<Cryptage-asymétrique rdf:ID="RSA"/>
<owl:ObjectProperty rdf:ID="réduit">
   <rdfs:domain rdf:resource="#Contremesure"/>
   <rdfs:range rdf:resource="#Vulnérabilité"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="s_exécute">
   <rdfs:domain rdf:resource="#Agent"/>
   <owl:inverseOf rdf:resource="#exécute"/>
   <rdfs:range rdf:resource="#Hôte"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Service">
   <rdfs:subClassOf rdf:resource="#Software"/>
</owl:Class>
<owl:Class rdf:ID="Signature-numérique">
   <rdfs:subClassOf rdf:resource="#Clé-de-cryptage"/>
</owl:Class>
<owl:Class rdf:ID="Société">
   <rdfs:subClassOf rdf:resource="#Propriétaire"/>
   <owl:disjointWith rdf:resource="#Personne"/>
</owl:Class>
<owl:Class rdf:ID="Software">
   <rdfs:subClassOf rdf:resource="#Asset"/>
</owl:Class>
<Protocole-gestion-clés rdf:ID="SSH"/>
<Protocole-gestion-clés rdf:ID="SSL">
   <assureObjectif rdf:resource="#Confidentialité"/>
</Protocole-gestion-clés>
<owl:Class rdf:ID="Storage">
   <rdfs:subClassOf rdf:resource="#Hardware"/>
</owl:Class>
<owl:Class rdf:ID="Token-physique">
   <rdfs:subClassOf rdf:resource="#Passeport"/>
</owl:Class>
<owl:Class rdf:ID="Token-électronique">
   <rdfs:subClassOf rdf:resource="#Passeport"/>
</owl:Class>
<Protection-physique rdf:ID="TPE"/>
<owl:Class rdf:ID="Trojan-horse">
   <rdfs:subClassOf rdf:resource="#Menace"/>
</owl:Class>
<Cryptage-symétrique rdf:ID="TwoFish"/>
<owl:DatatypeProperty rdf:ID="téléphone">
   <rdfs:domain rdf:resource="#Propriétaire"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="utiliseAlgorithme">
   <rdfs:domain rdf:resource="#Protocole"/>
   <rdfs:range rdf:resource="#Algorithme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="utiliseCryptage">
   <rdfs:domain rdf:resource="#Protocole-cryptage"/>
   <owl:inverseOf rdf:resource="#estUtiliséProtocoleCryptage"/>
   <rdfs:range rdf:resource="#Algorithme-cryptage"/>
```

```
      <rdfs:subPropertyOf rdf:resource="#utiliseAlgorithme"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="utiliseMécanisme">
      <rdfs:domain rdf:resource="#Protocole"/>
      <owl:inverseOf rdf:resource="#utiliseProtocole"/>
      <rdfs:range rdf:resource="#Mécanisme"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="utilisePasseport">
      <rdfs:domain rdf:resource="#Algorithme"/>
      <rdfs:range rdf:resource="#Passeport"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="utiliseProtocole">
      <rdfs:domain rdf:resource="#Mécanisme"/>
      <owl:inverseOf rdf:resource="#utiliseMécanisme"/>
      <rdfs:range rdf:resource="#Protocole"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="utiliseSignature">
      <rdfs:domain rdf:resource="#Protocole-signature"/>
      <owl:inverseOf rdf:resource="#estUtiliséProtocoleSignature"/>
      <rdfs:range rdf:resource="#Algorithme-signature"/>
      <rdfs:subPropertyOf rdf:resource="#utiliseAlgorithme"/>
   </owl:ObjectProperty>
   <owl:DatatypeProperty rdf:ID="valeur-cookie">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:domain rdf:resource="#Cookie"/>
      <rdfs:range rdf:resource="&xsd;int"/>
   </owl:DatatypeProperty>
   <owl:FunctionalProperty rdf:ID="version">
      <rdf:type rdf:resource="&owl;DatatypeProperty"/>
      <rdfs:domain>
         <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
               <owl:Class rdf:about="#Antivirus"/>
               <owl:Class rdf:about="#X.509"/>
            </owl:unionOf>
         </owl:Class>
      </rdfs:domain>
      <rdfs:range rdf:resource="&xsd;string"/>
   </owl:FunctionalProperty>
   <owl:Class rdf:ID="Virus">
      <rdfs:subClassOf rdf:resource="#Menace"/>
   </owl:Class>
   <owl:FunctionalProperty rdf:ID="vitesse">
      <rdf:type rdf:resource="&owl;DatatypeProperty"/>
      <rdfs:domain rdf:resource="#Microprocesseur"/>
      <rdfs:range rdf:resource="&xsd;int"/>
   </owl:FunctionalProperty>
   <owl:Class rdf:ID="Vulnérabilité"/>
   <owl:Class rdf:ID="Worm">
      <rdfs:subClassOf rdf:resource="#Menace"/>
   </owl:Class>
   <owl:Class rdf:ID="X.509">
      <rdfs:subClassOf rdf:resource="#Certificat"/>
   </owl:Class>
</rdf:RDF>
```