Agents ubiquitaires pour un accès adapté aux systèmes d'information : Le Framework PUMAS

Angela Cristina Carrillo Ramos

UNIVERSITE JOSEPH FOURIER

N°attribué par la bibliothèque

|__|__|__|__|__|__|__|__|__|__|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE JOSEPH FOURIER

Spécialité : Informatique

préparée au Laboratoire d'Informatique de Grenoble

dans le cadre de l'École Doctorale
Mathématiques, Informatique, Sciences et Technologie de l'Information

présentée et soutenue publiquement

par

Angela Cristina CARRILLO RAMOS

le 5 mars 2007

Agents ubiquitaires pour un accès adapté aux systèmes d'information : Le Framework PUMAS

Directeur de Thèse : Hervé MARTIN

JURY

M. Bruno Defude,	Président
M. Olivier Boissier,	Rapporteur
M. Lionel Brunie,	Rapporteur
M. Hervé Martin,	Directeur de Thèse
M. Jérôme Gensel	Co-encadrant
M. Gérôme Canals	Examinateur

À Fernando, mon amour, mon époux

À Luz Mila, ma mère

À Inés, ma grand-mère

REMERCIEMENTS

Je tiens à remercier :

Monsieur Bruno DEFUDE, Directeur d'études au département Informatique de l'Institut National des Télécommunications à Evry, de m'avoir fait l'honneur de présider le jury de ma thèse.

Monsieur Lionel BRUNIE, Professeur de l'Institut National des Sciences Appliquées de Lyon, et Monsieur Olivier BOISSIER, Professeur à l'Ecole Nationale Supérieure des Mines de Saint-Étienne, pour avoir accepté d'être les rapporteurs de ma thèse. Je les remercie pour les commentaires qu'ils ont apportés à mon mémoire de thèse.

Monsieur Gérôme CANALS, Maître de Conférences à l'Université de Nancy 2, pour avoir accepté de participer à mon jury de thèse comme examinateur.

Un remerciement très spécial à mes trois chefs :

Monsieur Hervé MARTIN, Professeur à l'Université Joseph Fourier, merci de m'avoir accueilli dans l'équipe STEAMER. Merci pour ton aide, tes commentaires toujours dans le moment précis et pour ta confiance dans mon travail. Merci de m'avoir mise dans les meilleures conditions pour développer PUMAS et le SGPC.

Monsieur Jérôme GENSEL, Maître de Conférences à l'Université Pierre-Mèndes France. Je te remercie de m'avoir appris à écrire des articles et mon mémoire de thèse. Merci pour ta disponibilité et pour les corrections de tous les documents que j'ai écrits pendant ma thèse. Tu m'as toujours fait confiance pendant ma thèse et tes commentaires m'ont permis d'améliorer mon travail.

Madame Marlène VILLANOVA-OLIVER, Maître de Conférences à l'Université Pierre-Mèndes France. Je te remercie pour ton intérêt dans mon travail, ta patience et ton aide pendant toute la thèse, notamment, pendant la rédaction. Merci aussi d'avoir introduit l'adaptation dans ma thèse. Merci pour tes encouragements et tes conseils. Je te remercie d'avoir toujours été à mon écoute et pour ton aide pendant les moments difficiles. Je me rappelle de ces longues journées de travail avec Jérôme et toi afin de réussir à faire de PUMAS et le SGPC une réalité.

Merci à tous les trois de m'avoir soutenu et encouragé pendant les périodes très difficiles qui me sont survenues. Avec votre soutien (de vous, mes trois chefs) j'ai pu rester en France et faire réalité mon rêve de devenir Docteur en Informatique.

Je garde dans ma mémoire les messages d'encouragement de tous les trois tant pour les acceptations des articles et pour aller chaque fois beaucoup plus loin que les mots de courage lors de la réception des mauvaises nouvelles. Vous m'avez appris à recevoir les critiques et chercher toujours l'amélioration de notre travail. Merci mes chefs !!!.

Je voudrais également remercier :

Les personnels administratifs et autres membres du Laboratoire d'Informatique de Grenoble pour leur constante amabilité et leur efficace assistance dans toutes mes démarches administratives.

L'Universidad de los Andes (Bogotá, Colombie) pour le financement partiel de mes études de doctorat.

Tous mes collègues de l'équipe SIGMA-STEAMER : Aurélie, Bogdan, Carlos, Céline, Christine, Christophe, Dia, Edicarsia, José, Manuele, Marius, Olivier, Raphaël, Thierry et Windson. Merci pour tous

les moments que nous avons partagé durant ma thèse. J'aurai toujours un bon souvenir de tous. Un remerciement très special à : Madame Paule-Annick DAVOINE, Maître de Conférences à l'INP de Grenoble pour sa gentillese et ses encouragements ; Céline LOPEZ-VELASCO pour son amitié, son aide et ses corrections de français pendant toutes ces années ainsi que notre idée de PUMAS-AWS ; Bogdan et Diana MOISUC pour leur amitié et leur bonne humeur ; Marius BILASCO et nos expériences avec la configuration des DM de notre équipe ; Windson VIANA, un collègue de bureau qui m'a beaucoup fait rire ; Raphaël THOMAS pour ses corrections de français de dernière minute et sa sympathie caractéristique ; Tati BARBIERO pour son amitié et son aide lors de mon arrivée au laboratoire ; Aurélie ARNAUD pour sa folie caractéristique et sa bonne humeur ; Thierry et Pascale BISSLER pour leur amitié et leurs cadeaux lors de mon arrivée ; Jorge PEREZ pour son aide au pot.

Mes amis de l'équipe HADAS avec qui j'ai beaucoup partagé surtout pendant les déjeuners qui sont devenus un moment de distraction et de joie : Thi Huong Giang VU (ma sœur vietnamienne), Hanh TAN (mon copain de cinéma), Laurent D'Orazio, Alberto PORTILLA (mon coloc mexicain) et Victor CUEVAS (merci de ton aide au pot).

Mes amis de Drakkar, spécialement Yan GRUNEMBERG (pour les tests sur le réseau Wi-Fi) et Vincent UNTZ (pour son aide avec le mystérieux « *Latex* » et la patience pour la mise en forme de l'article d'AP2PC).

Mes amis colombiens à Grenoble qui m'ont beaucoup encouragé et aidé : Astrid JAIME et Rodrigo SCIOVILLE✞.

Laura et Paolo SAVIOTTI pour leur amitié et aide durant tout mon séjour en France.

Les familles MENDEZ-LINDO (Isa, Trino, Rodolfo, Daniel, Athala et leurs enfants) et MOTTA MENDEZ (José Fernando, Fabian Enrique, Alexandra et Enrique). Vos encouragements depuis la Colombie m'ont beaucoup aidé dans les bons et les mauvais moments. Vous faites partie de ma famille et vous avez toujours été attentifs à mes nouvelles.

Mes amis Alexandra MENDEZ LINDO et Jeimy J. CANO MARTINEZ. Merci de vos encouragements chaque fois que j'en avais besoin. Merci d'être mes amis et de garder notre amitié malgré la distance.

Mes amis en Colombie pour vos messages d'encouragement : Olga Lucia, Julio, Nicolas, Alvaro, Tere, Juan Pablo et Nancy.

Ma belle famille, notamment ma belle mère Fanny ROSERO pour son aide et ses encouragements pendant toute ma thèse.

Les derniers remerciements mais les plus importants pour moi sont pour les trois personnes les plus importantes de ma vie :

Fernando, mon « *Mualito* ». Merci mon amour de m'avoir appuyé dans mon rêve de devenir Docteur en Informatique. Merci d'être venu en France pendant la première année de ma thèse et d'avoir continué tes voyages pour nous rencontrer toutes les vacances. Je te remercie d'avoir attendu mon arrivée tout seul chez nous et d'être en charge de notre foyer à Bogotá. Je te remercie de tout ton amour, de ton soutien, de ta patience et de ton aide pendant ces trois dernières années. Merci pour ta patience et ta disponibilité pour lire et faire des commentaires sur mon manuscrit malgré la distance et tout le travail que tu avais. Cette thèse est aussi une réalité grâce à toi, mon amour.

« *Mami Lucita* » et « *Bubu Inés* ». Merci ma mère et ma grand-mère d'être à côté de moi tout ce temps (soit ici à Grenoble, soit chez nous). Merci pour vos voyages, votre amour et soutien pendant ce processus. Merci d'avoir aussi soutenu Fernando en Colombie et de m'amener dans chacun de vos voyages une partie de ma Colombie (« *Colombia patria querida* »). Je vous remercie de vous être occupé de toutes les choses de la maison à Grenoble et de trouver un repas chaud à mon retour après une longue

journée de travail. J’appréciais tous les midis mon déjeuner qui venait d’être préparé. Votre aide a toute la valeur du monde.

Merci de votre aide inconditionnelle et d’être toujours à mes côtés. Que Dieu vous bénisse, mes anges gardiens.

TABLE DES MATIERES

1. INTRODUCTION 1
Contexte de ce travail 1
Problématique 2
Aperçu de la proposition 4
Organisation du document 6

ÉTAT DE L'ART 9

2. UBIQUITE, SYSTEMES PAIR A PAIR ET MOBILITE 11
2.1. Informatique ubiquitaire 11
2.2. Systèmes Pair à Pair 14
2.2.1 Classification de systèmes Pair à Pair 14
2.2.2 Exemples de systèmes Pair à Pair 15
2.2.3 Processus de routage de requêtes 16
2.2.3.1 Définition 17
2.2.3.2 Routage de requêtes dans les systèmes P2P 17
2.2.3.3 Traitement de requêtes 21
2.3. Requêtes basées sur la localisation 22
2.3.1 Requêtes basées sur la localisation 22
2.3.2 Mécanismes de capture de la localisation 23
2.4. Conclusion 23

3. SYSTEMES MULTI-AGENTS ET REPRESENTATION DE CONNAISSANCES POUR LES ENVIRONNEMENTS NOMADES 27
3.1. Définition des Systèmes Multi-Agents et caractéristiques des agents 29
3.2. Architectures des Systèmes Multi-Agents 31
3.3. Interaction entre agents 37
3.3.1 Langage pour la modélisation de SMA 38
3.3.2 Langages de communication 40
3.4. Représentation de connaissances 41
3.4.1 Ontologies 42
3.4.1.1 Définition 42
3.4.1.2 Langage de représentation 44
3.4.2 Règles 46
3.4.2.1 Classement 46
3.4.2.2 Langages de règles 46
3.5. Conclusion 48

4. ADAPTATION DANS DES ENVIRONNEMENTS NOMADES 51
4.1. Adaptation des Systèmes d'Information 52
4.1.1 Types d'adaptation 53
4.1.1.1 Adaptation au niveau du contenu 54
4.1.1.2 Adaptation au niveau de la navigation 54
4.1.2 Systèmes Hypermédia Adaptatifs 55
4.1.3 Modèles utilisés pour l'adaptation 57
4.2. Représentation du profil de l'utilisateur 58
4.3. Représentation des préférences de l'utilisateur 59
4.4. Représentation du contexte d'utilisation 60
4.5. Adaptation à l'aide d'agents 63
4.5.1 Architectures qui adaptent l'information 64
4.5.2 Techniques pour l'adaptation dans des Systèmes Multi-Agents 66
4.6. Conclusion 68

5. SYNTHESE 71
5.1. Agents pour l'accès aux Systèmes d'Information à travers des DM 71
5.2. Adaptation de l'information 73
5.3. Contribution de notre proposition 76

PROPOSITION 81

6. LE FRAMEWORK PUMAS 83
6.1. Architecture logique 85
6.1.1 L'architecture de PUMAS 86
6.1.1.1 Le SMA de connexion 87
6.1.1.2 Le SMA de communication 88
6.1.1.3 Le SMA d'information 89
6.1.2 Le SMA d'adaptation 91
6.1.3 Définition de profils et de règles pour les agents 93
6.2. Architecture fonctionnelle 93
6.2.1 Echange d'information pour l'adaptation 94
6.2.1.1 Echanges impliquant les agents du SMA de connexion 94
6.2.1.2 Echanges impliquant des agents du SMA de communication 95
6.2.1.3 Les agents du SMA d'adaptation 96
6.2.2 Envoi des requêtes de l'utilisateur 98
6.2.3 Un exemple d'utilisation de PUMAS 99
6.2.4 Représentation de connaissances 100
6.2.4.1 Les ontologies 100
6.2.4.2 Génération et gestion de bases de connaissances 105
6.3. Conclusion 110

7. PROCESSUS DE ROUTAGE DE REQUETES DANS PUMAS 113
7.1. Processus de routage de requêtes 113
7.1.1 Analyse de la requête 114
7.1.2 Sélection de Systèmes d'Information 117
7.1.3 Redirection de la requête 120
7.2. Impact des changements de localisation 120
7.3. Scénarios d'utilisation de PUMAS 121
7.3.1 Scénario associé à la connexion 121
7.3.2 Scénario d'envoi d'une requête d'information 122

7.3.3 Scénario de réception des résultats d'une requête d'information 124
7.3.4 Exemple 125
7.4. Conclusion 128

8. GESTION DES PREFERENCES DE L'UTILISATEUR 130
8.1. Formalisation 131
8.1.1 Préférences d'activité 132
8.1.2 Préférences de résultat 133
8.1.3 Préférences d'affichage 134
8.2. Définition du profil de l'utilisateur 135
8.2.1 Vue d'ensemble du processus 135
8.2.2 Définition du profil contextuel de l'utilisateur 135
8.2.3 Algorithme de correspondance contextuelle 137
8.3. Résolution de conflits entre préférences de l'utilisateur 140
8.4. Système de Gestion de Profils Contextuels 142
8.5. Exemple 143
8.6. Intégration de l'adaptation à PUMAS 149
8.7. Conclusion 151

9. MISE EN ŒUVRE DE PUMAS ET DE LA GESTION DES PREFERENCES DE L'UTILISATEUR 153
9.1. PUMAS – Implémentation 153
9.1.1 Modélisation des Systèmes Multi-Agents 153
9.1.2 Modélisation du Système de Gestion de Profil Contextuel 159
9.1.3 Protégé comme outil pour définir les ontologies 161
9.1.4 Plates-formes de développement 165
9.1.4.1 Spécifications *FIPA* pour les applications nomades et les plates-formes de développement 165
9.1.4.2 Plates-formes conformes aux spécifications de la FIPA 167
9.1.5 Correspondance entre les composants de PUMAS et les composants JADE 171
9.2. Exemples d'interfaces fournies à l'utilisateur/administrateur 172
9.2.1 Connexion de l'utilisateur à PUMAS 172
9.2.2 Envoi de messages entre utilisateurs 182
9.3. Conclusion 184

CONCLUSIONS 185

10. CONCLUSION ET PERSPECTIVES 187
10.1. Conclusion 187
10.2. Perspectives 192

11. REFERENCES 195

TABLE DES ILLUSTRATIONS

FIGURES

Figure 3.1. Architecture de CONSORTS (D'après Kurumatani et al. [Kuru04]). 34
Figure 3.2. Diagramme de séquence de flux envoyés de manière concurrente. 40
Figure 3.3. Diagramme de séquence de flux envoyés en tenant compte d'une condition. 40
Figure 3.4. Diagramme de séquence de flux envoyés en tenant compte d'une condition exclusive n'envoyant qu'un flux d'exécution. 40
Figure 4.1. Représentation multidimensionnelle du profil utilisateur (d'après Zemirli et al. [Zemi05]). 59
Figure 4.2. Diagramme de classes de la représentation de contexte d'utilisation d'un utilisateur nomade dans un collecticiel (D'après Kirsch-Pinheiro [Kirs06]). 62
Figure 4.3. Hiérarchie des entités et leurs propriétés (D'après Bucur [Bucu06]). 63
Figure 6.1. Architecture logique de PUMAS. 86
Figure 6.2. Les agents du SMA de connexion. L'agent de DM s'exécute sur le DM et l'agent contrôleur de connexions sur la plate-forme centrale de PUMAS. 87
Figure 6.3. Diagramme de classes d'un agent de DM qui possède les caractéristiques et opérations d'un agent mobile, d'un agent coopératif et d'un agent de connexion. 87
Figure 6.4. Communication entre les agents du SMA de communication. 88
Figure 6.5. Communication entre les agents du SMA d'information. 90
Figure 6.6. Le SMA d'adaptation. 92
Figure 6.7. Architecture logique de PUMAS contenant le SMA d'adaptation. 93
Figure 6.8. Echange d'information de l'agent d'utilisateur. 96
Figure 6.9. Echange d'information de l'agent de filtre d'affichage. 97
Figure 6.10. Echange d'information de l'agent de filtre de contenu. 98
Figure 6.11. Diagramme de séquence pour envoyer une requête d'information. 99
Figure 6.12. Un système d'emploi du temps géré par PUMAS. 100
Figure 6.13. Ontologie du profil de dispositif. 101
Figure 6.14. Ontologie de la session. 102
Figure 6.15. Ontologie de la localisation. 102
Figure 6.16. Représentation de la localisation en GML. 103
Figure 6.17. Hiérarchie de classes de GML. 103
Figure 6.18. Ontologie de l'utilisateur. 105
Figure 6.19. Définition d'un Template JESS. 106
Figure 7.1. Scénario associé à la connexion. 122
Figure 7.2. Scénario d'envoi d'une requête. 123
Figure 7.3. Scénario de réception des résultats d'une requête. 125
Figure 7.4. Envoi d'une requête dans le SI de l'hôpital. 126

Figure 8.1. Processus d'adaptation. 135
Figure 8.2. Préférences d'un utilisateur et préférences de substitution qui leur sont associées..... 139
Figure 8.3. Système de Gestion de Profils Contextuels (SGPC). 142
Figure 8.4. Processus de filtrage d'information à travers l'analyse des préférences de l'utilisateur. 149
Figure 9.1. Diagramme de packages de PUMAS. 154
Figure 9.2. SMA de connexion de PUMAS..... 155
Figure 9.3. SMA de communication de PUMAS. 155
Figure 9.4. SMA d'information de PUMAS..... 156
Figure 9.5. SMA d'adaptation de PUMAS..... 156
Figure 9.6. Exemple d'envoi concurrent de messages entre agents..... 157
Figure 9.7. Exemple d'exécution conditionnée d'envoie de messages entre des agents. 157
Figure 9.8. Diagramme de séquence AUML pour une requête d'information. 158
Figure 9.9. Diagramme d'état du traitement d'une requête. 158
Figure 9.10. Diagramme de classes de préférences d'activité, de résultat et d'affichage. 159
Figure 9.11. Diagramme de classes du profil contextuel d'un utilisateur..... 160
Figure 9.12. Interface de Protégé. Il est possible d'ouvrir un projet sous plusieurs formats (cf. la liste des boutons radio)..... 161
Figure 9.13. L'nterface de Protégé fournit différents plug-ins, par exemple, pour la gestion de connaissances (JessTab à gauche) et pour la visualisation d'ontologies (ezOWL à droite). 162
Figure 9.14. Interface de Protégé. Génération du code d'une ontologie en OWL..... 163
Figure 9.15. Interface de Protégé. Création d'une propriété..... 164
Figure 9.16. Interface de Protégé. Définition de la hiérarchie de classes et leurs propriétés. 165
Figure 9.17. Modèle de Référence pour la gestion des agents (d'après [fipa00023]). 166
Figure 9.18. Plate-forme JADE-LEAP avec les agents de base de la plate-forme : rma, ams et df, et l'agent preferencemanager (agent contrôleur de connexions) qui gère les connexions à PUMAS et les préférences de l'utilisateur. 174
Figure 9.19. Interfaces de trois utilisateurs simulés, chacun avec un agent de DM associé. La simulation a été développée avec JBuilder. Les frames supérieurs montrent l'interface de chaque utilisateur, les inférieurs correspondent aux utilisateurs pairs de chacun. . 175
Figure 9.20. Le trois agents créés en utilisant JBuilder s'ajoutent à la liste d'agents de la plate-forme. 175
Figure 9.21. L'utilisatrice « Angela » se connecte à PUMAS à travers le Pocket PC (à gauche). A droite, une interface de la plate-forme JADE-LEAP qui montre la création d'un Back-End pour chaque utilisateur qui se connecte à la plate-forme..... 176
Figure 9.22. Mise à jour de l'interface de la plate-forme JADE-LEAP avec l'agent correspondant à la nouvelle utilisatrice « Angela »...... 176
Figure 9.23. Mise à jour des interfaces des utilisateurs simulés en ajoutant l'utilisatrice « Angela » comme pair pour chacun. 177
Figure 9.24. Interface de l'utilisateur pour le Pocket PC. L'agent correspondant à l'utilisatrice « Angela » est connecté..... 177
Figure 9.25. Mise à jour de l'interface de l'utilisateur muni d'un Pocket PC. Les trois agents initialement enregistrés sur la plate-forme deviennent des pairs de l'agent correspondant à l'utilisatrice « Angela »..... 178
Figure 9.26. Mise à jour des interfaces des utilisateurs simulés en ajoutant l'utilisatrice « Céline » comme pair pour chacun. 179
Figure 9.27. Interface de l'utilisatrice « Céline » sur le Pocket PC. Ses agents pairs sont ceux simulés sur le serveur plus celui de « Angela » s'exécutant sur le même Pocket PC. 179

Figure 9.28. Le gestionnaire de tâches du Pocket PC montre les applications d'agents s'exécutant sur le même Pocket PC pour deux utilisateurs différents.....180
Figure 9.29. Interface de la plate-forme JADE-LEAP. L'agent de DM correspondant à l'utilisateur Raphaël s'est déconnecté. Bien que son agent proxy associé apparaisse comme « connecté » à la plate-forme, il n'apparaît plus sur l'interface des agents de DM. ...180
Figure 9.30. Interface de l'utilisateur pour le Pocket PC. L'agent de DM correspondant à l'utilisateur Raphaël s'est déconnecté. Il n'apparaît plus comme pair des agents de DM. 181
Figure 9.31. Interface de la plate-forme JADE-LEAP. Les agents proxy demandent à l'agent contrôleur de connexions tous les agents proxy qui se sont connectés à la plate-forme (c'est-à-dire, l'historique). L'agent contrôleur de connexions leur envoie une liste dans laquelle les agents déconnectés apparaissent avec leur adresse complète (pour les différencier des agents connectés).....181
Figure 9.32. Interface de l'utilisateur du Pocket PC. L'agent de DM correspondant à l'utilisatrice « Angela » demande tous les agents qui se sont connectés à la plate-forme. Les agents déconnectés apparaissent avec leur adresse complète.....182
Figure 9.33. Interface de la plate-forme JADE-LEAP et des frames des quatre utilisateurs connectés. 182
Figure 9.34. Interface de la plate-forme JADE-LEAP. Cette figure montre la plate-forme et les interfaces des agents de DM correspondant à quatre utilisateurs connectés. L'interface de l'agent de DM correspondant à l'utilisateur Raphaël montre que cet utilisateur a sélectionné les utilisateurs Fernando et Bogdan pour leur envoyer le message « Message pour Fernando et Bogdan ». 183
Figure 9.35. Interface de la plate-forme JADE-LEAP. Cette figure montre la plate-forme et l'interface des quatre utilisateurs connectés. L'utilisateur Windson n'a sélectionné aucun utilisateur. Son message a été transmis en broadcast. 184

TABLEAUX

Tableau 1. Architectures pour modéliser SMA. 36
Tableau 2. Architectures pour modéliser SMA. 72
Tableau 3. Défauts des trois niveaux communs pour les architectures basées sur des agents. 73
Tableau 4. Travaux qui représentent un profil de l'utilisateur selon plusieurs dimensions. 74
Tableau 5. Travaux basés sur des agents qui adaptent l'information dans des environnements nomades. 76
Tableau 6. Les profils générés par l'algorithme de correspondance contextuelle. 139
Tableau 7. Traitement dans PUMAS des aspects de base des architectures qui modélisent des SMA. 189
Tableau 8. Dimensions considérées dans notre représentation du profil de l'utilisateur. 190
Tableau 9. Aspects pris en compte par notre proposition basée sur des agents pour adapter l'information à l'utilisateur. 191

1. INTRODUCTION

Contexte de ce travail

Actuellement, l'Internet est utilisé de manière intensive pour accéder et échanger de l'information. A chaque accès, les utilisateurs souhaitent obtenir l'information la plus pertinente par rapport à leurs caractéristiques et à celles de leur(s) dispositif(s) d'accès. Cependant, l'utilisateur nomade (utilisateur qui change fréquemment de localisation) peut obtenir lors d'un accès, une grande quantité d'information qui n'est pas toujours pertinente, ni toujours supportée par son *Dispositif Mobile* (*DM*). Au cours de la dernière décennie, l'accès aux *Systèmes d'Information basés sur le Web* (*SIW*)[1] a beaucoup changé en raison de nombreux facteurs [Berh06] :

- La mobilité intrinsèque de l'utilisateur nomade ;
- Les avancées techniques dans le domaine des *DM*, par exemple *PDA*, téléphones, ordinateurs portables ;
- Le besoin de prendre en compte les capacités matérielles et logicielles réduites des *DM* (par exemple, taille de l'écran, mémoire, disque dur) ;
- La nature multimédia des données échangées.

Le changement sous-jacent consiste à fournir aux utilisateurs de *SIW* de l'information utile basée sur une *recherche intelligente de l'information* et un *affichage approprié* (en considérant les caractéristiques de l'utilisateur et celles de son *DM*).

Actuellement, les *DM* peuvent être utilisés afin d'accéder aux *SIW* distants mais aussi pour stocker des petites quantités d'information (par exemple, des fichiers) ou encore pour exécuter des applications ou des *SIW* simples[2]. Afin de permettre l'accès aux *SIW* et le partage de

[1] Selon Villanova [Vill02], les *Systèmes d'Information basés sur le Web* (*SIW*) sont des systèmes qui permettent de collecter, structurer, stocker, gérer et diffuser de l'information, de la même manière que les *Systèmes d'Information* (*SI*) traditionnels le font, mais via le Web. Un *SIW* fournit généralement aux utilisateurs des fonctionnalités complexes qui sont activées à travers un navigateur Web en utilisant une interface hypermédia.

[2] On entend par *SIW* simple un *SIW* qui s'exécute sur un *DM* et peut fournir des services tels que l'accès, la recherche, l'affichage et le stockage d'information à l'intérieur du *DM*.

ressources entre des utilisateurs dans des environnements nomades, il est nécessaire de disposer d'architectures et de technologies ayant un grand potentiel communicatif. Ce potentiel est une des caractéristiques principales des *systèmes Pair à Pair* (*P2P*).

Par ailleurs, les concepteurs de *SIW* doivent disposer de mécanismes et d'architectures afin de stocker, récupérer et délivrer efficacement l'information la plus pertinente, en proposant un accès à travers des *DM*, quels que soient le lieu et le moment. Cela est l'idée sous-jacente de l'*informatique ubiquitaire*. Les concepteurs doivent également tenir compte des capacités réduites de ce type de dispositifs (par exemple, la taille de l'écran, la mémoire, le disque dur) pour un affichage approprié sur le *DM* de l'utilisateur.

Parmi les aspects à considérer lors de l'adaptation de l'information à l'utilisateur nomade, nous mentionnons :

- Tout d'abord, les changements de localisation de l'utilisateur qui peuvent induire des changements en termes d'accès et de besoins d'information. La localisation de l'utilisateur peut être obtenue à l'aide d'un dispositif *GPS* ou d'une des méthodes proposées par Nieto-Carvajal *et al.* [Niet04] : le « *Signal Strength* », le « *SNMP* (*Simple Network Management Protocol*) », l'*Accès Local* à l'adresse *MAC* du point d'accès, entre autres ;
- Ensuite, les préférences d'un utilisateur, relatives aux activités qu'il souhaite accomplir dans le système, aux résultats attendus de ces activités et à la manière dont il souhaite les voir affichés sur son dispositif d'accès, peuvent être appliquées en fonction des caractéristiques contextuelles de la session en cours. Pour cette raison, un *SIW* doit disposer de données qui portent d'une part sur le *contexte d'utilisation*[3] et d'autre part, sur les *préférences de l'utilisateur*. Les données sur le *contexte d'utilisation* permettent notamment (mais pas seulement) d'obtenir une description des conditions (temporelles, spatiales, matérielles, *etc*.) dans lesquelles l'utilisateur accède au *SIW*. Les données sur les préférences visent à traduire ce que l'utilisateur souhaiterait obtenir du système par rapport à différents axes (les fonctionnalités, le contenu, l'affichage, *etc*.).
- Finalement, les caractéristiques et contraintes techniques du dispositif d'accès de l'utilisateur nomade peuvent produire des problèmes d'affichage de l'information. Il faut donc également tenir compte des préférences de l'utilisateur par rapport à l'affichage de l'information sur son dispositif d'accès.

Cette thèse s'intéresse principalement à deux aspects : d'une part, celui de l'accès aux *SIW* à travers des *DM*, et d'autre part, celui de l'adaptation de l'information en considérant le *profil de l'utilisateur* (composé spécifiquement des *préférences de l'utilisateur* qui peuvent être applicables en tenant compte du *contexte d'utilisation* de la session en cours) ainsi que les caractéristiques et contraintes techniques de son dispositif d'accès.

Problématique

Plusieurs problèmes fonctionnels et techniques doivent être considérés lors de la conception d'un *SIW*. Le système ne doit pas répondre seulement aux requêtes, il doit également considérer les préférences de l'utilisateur nomade, son historique dans le système et son *contexte d'utilisation* pour la session en cours.

[3] Dans notre travail, le *contexte d'utilisation* est constitué d'informations sur la localisation de l'utilisateur, les caractéristiques du *DM*, les droits d'accès de l'utilisateur et ses activités dans le système.

En ce qui concerne l'accès aux *SIW* à travers des *DM*, les concepteurs doivent tenir compte de l'hétérogénéité des dispositifs à travers lesquels les utilisateurs peuvent se connecter et accéder à l'information. De même, l'affichage de l'information sur les *DM* est à considérer car ces dispositifs sont caractérisés par des contraintes techniques en rapport avec la performance, l'autonomie, le traitement et le stockage de données, entre autres. Malgré les progrès concernant l'infrastructure de connexion et de communication des *DM*, des problèmes subsistent. A ce jour, cette infrastructure ne peut ainsi pas garantir une connexion illimitée de l'utilisateur, les mêmes caractéristiques de réseau (par rapport à la bande passante, à la couverture, *etc*.), la disponibilité des données, leur transmission, *etc*.

Par ailleurs, en considérant l'adaptation, la localisation de l'utilisateur nomade (ainsi que son profil et les caractéristiques contextuelles de la session en cours) est devenue très importante car elle permet d'obtenir uniquement l'*information la plus appropriée* à son *contexte d'utilisation*. Thilliez *et al*. [Thie04] ont proposé les requêtes « *dépendantes de la localisation* » qui sont évaluées en tenant compte de la localisation physique de l'utilisateur.

Un autre aspect à prendre en compte est celui du traitement des requêtes des utilisateurs nomades. Ce traitement est devenu un processus de plus en plus complexe car le résultat des requêtes peut provenir de différents systèmes d'information et l'utilisateur ne souhaite obtenir que l'information la mieux adaptée à ses caractéristiques et à celles de son dispositif d'accès. Pour cette raison, il est nécessaire :

- d'analyser les requêtes afin de connaître les *systèmes d'information* qui peuvent y répondre ;
- de sélectionner les *systèmes d'information* les plus appropriés pour y répondre ;
- d'évaluer et d'intégrer leurs résultats ;

Ces activités correspondent à un processus traditionnel de *routage de requêtes*. D'après Xu *et al*. [XuLi99], le processus de *routage de requêtes* comprend deux activités principales : d'une part, l'évaluation d'une requête en utilisant les *systèmes d'information* les plus appropriés pour y répondre, et d'autre part, l'intégration des résultats retournés. Cependant, ces activités ne traitent pas explicitement de l'adaptation de l'information. Une augmentation de la requête avec l'information sur les *préférences de l'utilisateur*, les caractéristiques et contraintes de son dispositif d'accès, et les caractéristiques de son *contexte d'utilisation* de la session en cours, pourrait alors être envisagée et mise en œuvre afin de fournir à l'utilisateur l'information la mieux adaptée à ses besoins, ses préférences et son *contexte d'utilisation*.

Enfin, les concepteurs de *SIW* doivent également disposer de mécanismes et d'outils qui permettent la représentation du *profil de l'utilisateur* et celle des éléments qui le composent. Parmi les composants du *profil de l'utilisateur*, nous trouvons les *préférences de l'utilisateur*. Cependant, d'après nos recherches, il n'existe pas de représentation unifiée permettant d'exprimer les activités que l'utilisateur souhaite mener dans le système, les résultats attendus de ces activités et la manière dont les résultats devraient être affichés sur son dispositif d'accès. Nous n'avons pas trouvé non plus de mécanismes permettant de générer le *profil de l'utilisateur* en se basant sur les *préférences de l'utilisateur* et les caractéristiques du *contexte d'utilisation* de la session en cours.

Aperçu de la proposition

Afin de résoudre la problématique énoncée ci-dessus, notre proposition se situe à l'intersection de trois domaines différents, l'*informatique ubiquitaire*, les *systèmes multi-agents* et l'*adaptation de l'information*, pour les raisons suivantes :

- L'*informatique ubiquitaire* permet aux utilisateurs de consulter les données n'importe quand, n'importe où, à travers des *DM*. L'information obtenue par cette consultation peut provenir d'un ou plusieurs *systèmes d'information*. Afin de permettre aux utilisateurs de partager leurs ressources, il est nécessaire de disposer d'architectures et de technologies ayant un grand potentiel communicatif. Ces architectures et technologies sont celles sur lesquelles reposent les *systèmes Pair à Pair* (*P2P*). Ces systèmes permettent aux utilisateurs de formuler des requêtes pouvant être dirigées vers différents *systèmes d'information* pour obtenir l'information recherchée. Dans des environnements nomades, les requêtes et les besoins d'information d'un utilisateur peuvent être sensibles à sa localisation physique.
- Les *Systèmes Multi-Agents* (*SMA*) regroupent des entités intelligentes (les « *agents* ») capables de représenter l'utilisateur dans un *système d'information*, de communiquer avec d'autres agents ou avec l'utilisateur afin d'accomplir leurs tâches et de prendre l'initiative pour exécuter des activités lorsque certaines conditions sont réunies. Les *SMA* s'approchent des *systèmes P2P* par de nombreux aspects, tels que : un agent peut accomplir de manière autonome ses tâches ainsi que communiquer avec d'autres indépendamment d'un serveur et d'autres agents ; un agent peut également partager des ressources pour exécuter des tâches simples ou d'autres plus complexes à l'aide d'autres agents. Plusieurs travaux basés sur des agents ont été développés pour faciliter l'accès à l'information à travers des *DM*. Parmi ces travaux, nous avons identifié trois niveaux communs : un niveau *d'agents mobiles*, un niveau *intermédiaire* et un niveau de *système d'information*. Cependant, ces travaux ne tiennent pas compte des caractéristiques de l'utilisateur, ni de celles de son dispositif d'accès afin d'adapter l'information.
- L'*adaptation de l'information* peut porter sur le *contenu* et la *navigation* et prendre en compte les préférences et les caractéristiques de l'utilisateur. Un exemple de systèmes qui exploitent ces deux niveaux sont les *systèmes hypermédias adaptatifs*. Néanmoins, ces systèmes ne considèrent pas la localisation de l'utilisateur, ni les caractéristiques de son dispositif d'accès afin d'adapter l'information. Pour cette raison, nous nous sommes intéressés aux travaux reposant sur une approche à base d'agents qui adaptent l'information aux caractéristiques de l'utilisateur et/ou à celles de son dispositif d'accès.

Parmi les aspects techniques et fonctionnels qui doivent être considérés lors de la conception d'un *SIW* accédé à travers des *DM*, nous considérons plus spécialement ceux qui traitent du problème de l'adaptation de l'information délivrée à l'utilisateur nomade. Le but de notre travail est de fournir l'information la plus appropriée à ce type d'utilisateurs en tenant compte de leurs préférences ainsi que des caractéristiques contextuelles et de celles de leurs *DM*. Notre approche est basée sur la technologie multi-agents.

Tout d'abord, nous proposons *PUMAS* (***P**eer **U**biquitous **M**ulti-**A**gent **S**ystem*), un *framework* qui récupère l'information distribuée entre plusieurs *SIW* et/ou accédée à travers différents types de *DM*. Les objectifs de notre *framework* sont :

- de garantir à l'utilisateur nomade un accès à des *SIW* à travers différents types de *DM* ;

– d'adapter l'information à travers un processus de filtrage d'information. Ce processus est accompli par deux filtres : le *filtre de contenu* qui procède à la sélection de l'information la plus appropriée pour l'utilisateur par rapport à ses caractéristiques (l'information telle que son profil, sa localisation, ses dernières requêtes), et le *filtre d'affichage* qui transforme l'information fournie par le *filtre de contenu* en considérant les capacités techniques de son *DM*.

Dans *PUMAS*, les utilisateurs munis de *DM* communiquent entre eux grâce à une architecture *P2P hybride*. Nous avons suivi les recommandations de Shizuka *et al.* [Shiz04] qui considèrent que l'utilisation d'une architecture *P2P Hybride* est appropriée pour :

– la prévention des problèmes de sécurité liés à la mobilité des agents ;
– la communication point à point ou à travers une *diffusion de* « *un vers N* » (« *broadcast* ») entre les agents ;
– la gestion des états des agents (par exemple, « connecté », « déconnecté », « tué », *etc.*) et de leurs services fournis.

Suivant l'approche *P2P*, un *SIW basé sur des Agents* (*SIWA*) doit représenter la connaissance requise par chaque agent pour accomplir ses tâches associées aux différents rôles qu'il peut jouer (par exemple, client, serveur, coordinateur). Le travail de Panti *et al.* [Pant02] est un exemple d'un *Système Multi-Agents* (*SMA*) composé d'*agents pairs* dont le processus de définition de tâches est celui que nous avons utilisé dans notre approche. Ce processus permet de définir les activités des agents et leurs plans d'exécution. Les *agents pairs* peuvent échanger de l'information afin d'accomplir leurs tâches (d'une manière individuelle ou coopérative) mais également fournir leurs services aux utilisateurs ou à d'autres agents. Dans *PUMAS*, les utilisateurs équipés de *DM* peuvent être représentés par des agents pairs puisque *PUMAS* fournit des mécanismes qui leur permettent, d'une part, de connaître d'autres pairs avec lesquels travailler ou auxquels demander de l'information, et d'autre part, de communiquer avec d'autres (agents ou utilisateurs) afin d'échanger de l'information. *PUMAS* représente les données, les rôles des agents, et l'échange de données en utilisant des *fichiers* écrits en *OWL*.

Les utilisateurs peuvent également utiliser *PUMAS* pour formuler leurs requêtes. Les agents de *PUMAS* sont en charge d'adapter l'information en considérant le *profil de l'utilisateur* (composé de ses *préférences* applicables en fonction du *contexte d'utilisation* de la session en cours) ainsi que les capacités techniques de leurs *DM*. Les agents de *PUMAS* enrichissent les requêtes de l'utilisateur par l'information contenue dans son profil, et par les caractéristiques de son *DM*. Ils redirigent alors les requêtes de l'utilisateur vers différents *SIW* contenant l'intégralité ou une partie de l'information recherchée. Ces *SIW* peuvent s'exécuter sur différents serveurs ou *DM*.

En ce qui concerne l'adaptation de l'information, nous proposons une approche qui repose sur la génération d'un *profil contextuel de l'utilisateur*. Un tel profil est composé des *préférences d'un utilisateur* applicables en fonction du *contexte d'utilisation* courant. Notre approche contribue à l'adaptation de différentes manières :

– **La formalisation de la notion de *préférence de l'utilisateur***, offrant un support à la représentation de trois types de *préférences de l'utilisateur* : *activité, résultat et affichage* ;
– **L'utilisation des caractéristiques du *contexte d'utilisation* de la session en cours** pour la sélection des préférences de l'utilisateur, composant du profil de l'utilisateur. Ce contexte est constitué de l'information sur la localisation de l'utilisateur, les caractéristiques du *DM*, les droits d'accès de l'utilisateur et ses activités ;

- **Un algorithme de *correspondance contextuelle*** qui génère un *profil contextuel* pour un utilisateur donné, en analysant chaque *préférence utilisateur* afin d'évaluer si cette dernière peut être satisfaite par rapport au *contexte d'utilisation*. Ce *contexte* permet au système de ne sélectionner que les *préférences* qui sont compatibles avec les *activités* d'un utilisateur. Les préférences retenues sont les composantes du *profil contextuel de l'utilisateur*[4] ;
- **La gestion de conflits de préférences qui peuvent survenir**. Nous avons ainsi identifié certains *conflits-types*, leurs causes et la manière dont le système doit réagir afin de les résoudre, ou du moins, informer l'utilisateur de l'existence de ces conflits.

Organisation du document

Ce document est structuré en deux parties, correspondant respectivement à l'état de l'art et à notre proposition.

L'état de l'art est composé de quatre chapitres concernant les concepts liés à l'*Informatique Ubiquitaire* (chapitre 2), les *Systèmes Multi-Agents* (chapitre 3), l'*Adaptation* dans des environnements nomades (chapitre 4), et finalement, une synthèse (chapitre 5) dont l'objectif est de lier les concepts des trois chapitres précédents en soulignant les avantages et les inconvénients des travaux présentés dans chaque chapitre.

Dans le chapitre 2, nous présentons les concepts liés à l'*Informatique Ubiquitaire* qui permet aux utilisateurs de demander de l'information n'importe quand, n'importe où, à travers des *DM*. Nous présentons aussi les systèmes *P2P* fournissant des architectures et des technologies de communication pour l'*informatique ubiquitaire*. Nous abordons le processus de *routage de requêtes* permettant d'analyser les requêtes de l'utilisateur, de sélectionner les *systèmes d'information* les plus appropriés pour y répondre, et de rediriger les requêtes vers ces *systèmes d'information*. Enfin, nous mettons en évidence l'importance de la localisation de l'utilisateur nomade dans le traitement des requêtes puisque ces changements de localisation peuvent modifier les besoins d'information (surtout si les requêtes sont dépendantes de la localisation).

Dans le chapitre 3 portant sur des *Systèmes Multi-Agents* et sur la *représentation de connaissances* pour les environnements nomades, nous présentons : *i)* les caractéristiques principales des agents, *ii)* des architectures basées sur des agents permettant la recherche d'information, *iii)* les activités d'interaction entre les agents (telles que la coordination, le contrôle, la coopération et la négociation) nécessaires pour accomplir une tâche commune, et *iv)* les langages de description des interactions et des messages entre les agents. Finalement, nous abordons des méthodes et des techniques pour la représentation et la gestion de la connaissance des agents. Nous présentons les ontologies comme un moyen de représenter des connaissances et des règles pour leur gestion.

Le chapitre d'*adaptation* de l'information dans des environnements nomades présente une définition de l'adaptation, ses types (du contenu et de la navigation) et quelques modèles utilisés afin d'adapter l'information. Il présente aussi la représentation du profil de l'utilisateur, des travaux sur la gestion des préférences de l'utilisateur, la définition et la gestion du *contexte d'utilisation*, et finalement, des travaux basés sur des agents qui adaptent l'information à l'utilisateur en considérant ses caractéristiques et celles de son dispositif d'accès.

[4] Le *profil contextuel de l'utilisateur* ne recouvre donc ici que des préférences.

Enfin, le chapitre de synthèse évalue dans quelle mesure les travaux présentés dans l'état de l'art contribuent à l'accès et à l'adaptation de l'information dans des environnements nomades. Ce chapitre présente une analyse sur l'utilisation d'agents pour l'accès aux *SIW* à travers des *DM*, une analyse des travaux liés à la représentation des *préférences de l'utilisateur*, de son *contexte d'utilisation*, et une analyse des travaux basés sur des agents qui adaptent l'information dans des environnements nomades.

Notre proposition est organisée en quatre chapitres :

Tout d'abord, nous proposons *PUMAS*, notre *framework* pour l'accès aux *SIW* à travers des *DM*. Notre *framework* adapte l'information à l'utilisateur nomade en considérant ses caractéristiques et préférences, et les caractéristiques et contraintes de son dispositif d'accès. Le chapitre 6 présente l'architecture logique de *PUMAS* en décrivant chaque *SMA* et le besoin d'étendre l'architecture de base d'un *SIWA* avec un *SMA* en charge de l'adaptation. Enfin, nous décrivons l'architecture fonctionnelle de *PUMAS*, en présentant l'échange d'information à des fins d'adaptation et de représentation de connaissances.

Le chapitre 7 est consacré à la définition du processus de *routage de requêtes* dans *PUMAS* qui se base sur l'analyse de la requête et sa redirection vers les *systèmes d'information* capables d'y répondre. Nous montrons également l'impact des changements de localisation pour les requêtes sensibles à cet aspect. Enfin, nous illustrons l'utilisation de *PUMAS* à travers trois scénarios : la connexion d'un utilisateur, l'envoi d'une requête et la réception de ses résultats.

Dans le chapitre 8, nous proposons une représentation et un processus de gestion des *préférences d'un utilisateur nomade*. Ces préférences concernent les activités que l'utilisateur souhaite exécuter dans le système, les résultats attendus de ces activités et la manière d'afficher ces résultats sur son dispositif d'accès. Ensuite, nous montrons la manière dont le *profil contextuel* d'un utilisateur est généré à l'aide de l'algorithme de *correspondance contextuelle* en tenant compte des *préférences de l'utilisateur* et du *contexte d'utilisation* de la session en cours (composé par exemple, de la localisation et du moment de connexion). Cet algorithme analyse chaque préférence en sélectionnant celles qui peuvent être appliquées en considérant le *contexte d'utilisation*. Puis, nous présentons un mécanisme de gestion des conflits qui peuvent survenir entre les *préférences de l'utilisateur* constitutives du *profil contextuel* obtenu. Le responsable de l'exécution de l'*algorithme de correspondance contextuel* est le *Système de Gestion de Profils Contextuels* qui est également chargé d'obtenir les *préférences de l'utilisateur* et le *contexte d'utilisation*. Finalement, nous intégrons le processus de gestion de préférences et de génération de profils contextuels dans *PUMAS*.

Le chapitre 9 présente certains aspects de modélisation et d'implémentation de notre *framework PUMAS* ainsi que la modélisation du *Système de Gestion de Profils Contextuels*. Nous présentons des outils tels que *Protégé* et *JADE-LEAP* que nous avons choisis respectivement pour la création des ontologies et la gestion de connaissances, et pour l'exécution d'agents. Nous décrivons brièvement les plates-formes des *SMA* qui suivent les spécifications *FIPA* en soulignant la raison pour laquelle nous avons choisi *JADE-LEAP* pour l'implémentation de *PUMAS*. Finalement, nous illustrons des interfaces pour l'accès à la plate-forme centrale de *PUMAS* à travers des *DM* et pour l'envoi de messages.

Enfin, le chapitre 10 conclut ce document, par le rappel des contributions et la mise en évidence des perspectives de ce travail par rapport à notre *framework PUMAS*, à la représentation et la gestion de *préférences de l'utilisateur* ainsi qu'à la définition du *profil de l'utilisateur* en considérant ces préférences et le *contexte d'utilisation* de la session en cours.

ÉTAT DE L'ART

2. UBIQUITE, SYSTEMES PAIR A PAIR ET MOBILITE

Dans ce chapitre, nous présentons les concepts liés à l'*informatique ubiquitaire,* un paradigme récent de l'*informatique personnelle* dont l'objectif est de permettre aux utilisateurs de consulter des données n'importe quand, n'importe où. Ce nouveau mode de consultation nécessite des mécanismes de communication entre les utilisateurs et une ou plusieurs sources d'information, besoin auquel peuvent répondre les systèmes *P2P*. Ces systèmes reposent, en effet, sur des architectures et des technologies ayant un grand potentiel communicatif, support de l'informatique ubiquitaire, et permettant aux pairs de rester accessibles (et donc d'être réellement ubiquitaires) malgré leurs changements de localisation. Nous présentons les caractéristiques majeures des systèmes *Pair à Pair* (*P2P*). Puis, nous abordons de manière plus détaillée le routage de requêtes dans des systèmes *P2P* qui permet de chercher les sources d'information les plus appropriés pour répondre aux requêtes de l'utilisateur. Enfin, nous terminons ce chapitre par une section qui vise à mettre en évidence pour l'*information ubiquitaire,* l'importance des informations de localisation dans le traitement des requêtes.

2.1. Informatique ubiquitaire

De nos jours, les applications informatiques doivent permettre à leurs utilisateurs de consulter des données n'importe quand, n'importe où, à travers leurs dispositifs d'accès. C'est l'idée sous-jacente de l'*informatique ubiquitaire.* Le *W3C* [W3Cb05] définit l'*informatique ubiquitaire,* comme un paradigme émergeant de l'*informatique personnelle* (« *Personal Computing* ») caractérisée par :

– L'utilisation de dispositifs légers, manipulables et sans fil ;

- La nature pervasive[5] et sans fil des dispositifs qui demande des architectures réseaux supportant une configuration automatique et *ad hoc* ;
- La haute distribution, hétérogénéité, mobilité et autonomie de l'environnement.

Koch *et al.* [Koch04] mentionnent plusieurs changements introduits par *l'informatique ubiquitaire*. En premier lieu, on trouve un *changement d'infrastructure* qui nécessite la construction de technologies robustes tant matérielles que logicielles facilitant la connectivité mobile, l'identification de la localisation, la découverte de services, la tolérance aux fautes, *etc*. Egalement, on peut évoquer le *changement de services* qui concerne la manière dont l'infrastructure disponible est utilisée afin de fournir de nouveaux services à l'utilisateur.

A propos des dispositifs d'accès propres à l'*informatique ubiquitaire* (c'est-à-dire, des *Dispositifs Mobiles, DM*), Rahwan *et al.* [Rahw04] soulignent certaines de leurs caractéristiques et contraintes techniques telles que : *i)* le stockage limité ; *ii)* la puissance de traitement limitée (sur un *DM*, ne sont possibles que l'utilisation de petites bibliothèques et l'exécution de petites applications) ; *iii)* l'hétérogénéité de la représentation des données (cf. section 3.4.1.1) ; *iv)* le manque de standards de spécification de plans d'exécution de tâches (ce qui réduit les possibilités de partager les données et de communiquer entre des applications) ; *v)* le besoin d'adjoindre des composants matériels embarqués (pour les dispositifs d'accès, par exemple, les capteurs de lumière, de son, de localisation, des lecteurs de code barres) pour la prise en compte des caractéristiques contextuelles de l'utilisateur (l'information sur les changements des caractéristiques contextuelles doit être fournie par l'utilisateur ou doit être détectée d'une manière automatique, par exemple, la localisation peut être détectée à travers un capteur *GPS*). Compte tenu des caractéristiques des *DM*, Hristova *et al.* [Hris04] donnent quelques conseils pour concevoir et développer des applications exécutées sur des *DM* : concevoir des clients légers, utiliser des logiciels dont l'exécution mobilise peu de ressources mémoire, utiliser des mécanismes minimisant la latence et améliorant la performance de la transmission des données sur des réseaux sans fil, définir un plan d'exécution des applications (et leurs tâches) sur le *DM*, concevoir des interfaces simples, *etc*.

Une application exécutée sur des *DM* doit posséder la capacité d'une part, d'interagir avec l'utilisateur et, potentiellement, avec d'autres applications et, d'autre part, de raisonner sur les objectifs en termes d'utilisation de l'utilisateur nomade et la manière dont il peut les atteindre [Rahw04]. De telles applications demandent des architectures réseaux capables de supporter une configuration automatique et compatible avec les caractéristiques de l'*environnement de l'informatique ubiquitaire* telles que l'hétérogénéité, la mobilité, l'autonomie, la haute distribution, *etc*. De tels *environnements* sont définis par Pirker *et al.* [Pirk04] comme étant un réseau dynamique distribué de dispositifs et systèmes embarqués qui peuvent interagir avec des humains afin de satisfaire leurs besoins et de leur fournir une variété de services d'information, de communication, et de collaboration.

[5] Le terme « *pervasif* » fait référence à la définition d'*informatique pervasive*. Celle-ci fournit un accès approprié à l'information et aux applications à travers des dispositifs qui possèdent la capacité de fonctionner au moment et à l'endroit nécessaires [Agos00].

Un utilisateur nomade est souvent situé dans un environnement[6] très dynamique. Les utilisateurs peuvent changer d'endroit ce qui éventuellement peut modifier l'exécution de leurs tâches en fonction des caractéristiques contextuelles de l'endroit où ils se trouvent (par exemple, l'utilisateur peut disposer de conditions de réseau différentes, ou il peut changer de dispositif d'accès selon sa localisation). Dans un environnement nomade, la connectivité du réseau ne peut pas être assurée à tout moment à tout endroit, et ses caractéristiques contextuelles peuvent modifier les besoins et préférences de l'utilisateur (par exemple, l'utilisateur peut préférer obtenir l'information demandée sous forme d'image à la place d'une vidéo à cause de la vitesse de transmission). Dans ces conditions, Calisti *et al.* [Cali04] proposent d'exploiter des paramètres d'accès de réseau tels que la bande passante, les délais et le temps moyen d'accès, pour déterminer la « *meilleure* » chaîne d'accès possible compte tenu des politiques des fournisseurs d'accès, des préférences de l'utilisateur et des caractéristiques du dispositif d'accès, des besoins de l'application s'exécutant sur le dispositif d'accès, *etc*.

Plusieurs applications de l'*informatique ubiquitaire* tiennent compte des caractéristiques contextuelles et de celles de l'utilisateur afin de leur adapter l'information. Quelques unes de ces applications de l'*informatique ubiquitaire* sont présentées dans la suite.

MyCampus [Gand04] est une infrastructure basée sur le Web Sémantique, sensible au contexte, en particulier aux préférences de confidentialité de l'utilisateur (« *privacy preferences* »). *Mycampus* est utilisée afin d'afficher les informations journalières d'un campus universitaire (par exemple, les activités pour des étudiants, le calendrier d'activités, l'emploi du temps, la localisation à l'intérieur du campus, *etc*.). Son architecture a été étendue vers différents environnements tels que des applications professionnelles du bureau, des visites guidées, *etc*.

Des applications telles que *WAY* (« *Where Are You* ») [Coll03], *Ad-Me* (« *Advertising for mobile e-commerce user* ») [Hris04] et *Gulliver's Genie* [OHar02] ont été développées afin d'adapter l'information à la localisation de l'utilisateur, compte tenu des caractéristiques de son *DM* (en général, des *PDA* et des téléphones portables). *WAY* [Coll03] est un système pour la synchronisation et la prise de rendez-vous entre utilisateurs nomades à l'aide de leurs *PDA*. *Ad-Me* [Hris04] est un système créé pour l'office du tourisme de Dublin qui fournit aux utilisateurs nomades des services de publicité (des magasins, des restaurants) à partir de leur localisation et de leurs préférences. *Gulliver's Genie* [OHar02] est un guide touristique qui se configure selon la localisation et l'orientation de déplacement de l'utilisateur et personnalise l'information tenant compte de son profil (par exemple, *Gulliver's Genie* fait un plan de la ville en soulignant tous les musées si les sites touristiques les plus visités par l'utilisateur sont des musées).

L'*informatique ubiquitaire* a besoin d'architectures et de technologies dotées de fortes capacités de communication. Shizuka *et al.* [Shiz04] considèrent que les systèmes *Pair à Pair*

[6] Un *environnement nomade* peut être représenté en termes d'information contextuelle, telle que le moment de connexion, la localisation, les caractéristiques du *DM*, et les tâches de l'utilisateur, entre autres.

(*P2P*) sont une bonne solution, permettant que les pairs soient ubiquitaires malgré les changements de localisation. La section suivante leur est consacrée.

2.2. Systèmes Pair à Pair

Stuckenschmidt *et al.* [Stuc06] et Xu *et al.* [XuZh03] définissent l'Informatique *Pair à Pair* (« *Peer to Peer* », *P2P*) comme une forme d'informatique distribuée qui implique un grand nombre de *nœuds* (les *pairs*) informatiques autonomes et coopératifs qui communiquent et partagent des ressources et des services disponibles, en l'absence de tout contrôle centralisé [Shiz04]. De tels systèmes sont hautement dynamiques : les nœuds rejoignent ou quittent le système et changent leurs contenus et de rôles constamment.

D'après Zhuge *et al*. [Zhug04], un *pair* peut jouer le rôle de *serveur* lorsqu'il fournit des données, de l'information et des services, de *médiateur* lors de l'acheminement des besoins de requêtes, et de *client* lorsqu'il accède à l'information détenue par d'autres pairs.

Le partage de données entre pairs dans les systèmes *P2P* est considéré comme un problème car les pairs peuvent s'exécuter sur des plates-formes hétérogènes. Le fait que toutes les applications *P2P* utilisent la même plate-forme afin de partager les ressources est en effet peu probable [XuZh03]. Une solution possible à ce problème est un « *middleware*[7] *P2P* » [XuZh03] devenant une couche de logiciels distribués qui permet de rendre abstraite la complexité et l'hétérogénéité sous-jacentes des systèmes *P2P*. Par exemple, Xu *et al*. [XuZh03] proposent un *middleware P2P* (appelé « *PairBus* ») permettant l'interopérabilité d'applications qui s'exécutent sur différentes plates-formes. *PairBus* permet l'interopérabilité de différents systèmes *P2P* ainsi que l'échange entre des utilisateurs de différents systèmes *P2P* ou plates-formes partageant des ressources.

Dans la section suivante, nous présentons une classification de systèmes *P2P* tenant compte des ressources partagées par les pairs et de leurs fonctionnalités.

2.2.1 Classification de systèmes Pair à Pair

Stuckenschmidt *et al.* [Stuc06] distinguent trois catégories de systèmes *P2P* selon le type de ressources partagées :

- des applications où les pairs partagent des ressources de calcul (aussi nommées « *Grid Computing* ») ;
- des applications où les pairs partagent la logique d'application (aussi nommées « *Architectures basées sur des services* ») ;

[7] Selon Xu *et al.* [XuZh03] un « *middleware* » est une collection de services informatiques distribués qui permettent de faire communiquer des clients et des serveurs, s'exécutant sur des plates-formes hétérogènes, d'une manière flexible et correcte. Le rôle d'un « *middleware* » est de faciliter les tâches de conception, de construction et de gestion d'applications distribuées en fournissant des environnements de construction des applications réparties qui soient simples, cohérents et intégrés. L'élément clé d'un *middleware* est l'interopérabilité.

– des applications où les pairs partagent de l'information (aussi nommées des « *Systèmes P2P classiques* »).

Le travail présenté ici se situe dans le cadre de *systèmes P2P classiques*. Les *systèmes P2P classiques* sont généralement divisés en deux groupes [XuZh03] [Kolo04] : les *systèmes P2P non structurés* et les *systèmes P2P structurés* :

– *Les systèmes P2P non structurés* comprennent les systèmes *P2P Purs* et les systèmes *P2P Hybrides*. Dans un système *P2P Pur*, tous les pairs sont égaux et il n'y a pas de fonctionnalité centralisée. Chaque pair dispose d'une liste de pairs avec lesquels il peut communiquer, travailler et partager de l'information. Cependant, si c'est nécessaire, il est également capable de contacter d'autres pairs (par exemple, en consultant l'annuaire des pairs du système). Dans les systèmes *P2P Purs*, les messages sont envoyés sans la médiation d'un serveur. Dans un *système P2P Hybride*, les pairs sont organisés en deux couches : la couche supérieure correspond aux « *super pairs* » et la couche inférieure correspond aux *pairs communs*. Chaque *pair commun* peut se connecter avec un ou plusieurs *super pairs*. Les *super pairs* jouent le rôle de serveur de répertoire afin de faciliter la localisation de ressources. Un pair peut se connecter à un serveur indexé et à d'autres pairs. Dans cette connexion, quelques messages de gestion sont passés via le serveur et d'autres messages sont directement échangés entre pairs.
– *Les systèmes P2P structurés* possèdent un registre des données partageables ainsi qu'une fonction de « *hashing* » donnant la localisation de chaque pair dans le même espace d'identification. Les données comme les pairs possèdent un identificateur unique (ou clé). Afin de délivrer efficacement les requêtes au pair contenant les données souhaitées, chaque pair maintient un tableau de routage prédéfini ou créé sur la base des interactions précédentes avec d'autres pairs.

2.2.2 Exemples de systèmes Pair à Pair

Plusieurs travaux concernant les systèmes *P2P* ont été développés pour différentes activités telles que le traitement de requêtes, la gestion de la connaissance et l'échange d'information à l'intérieur d'un groupe de travail. Nous présentons des exemples de systèmes *P2P* pour chacune de ces activités :

Vdovjak *et al*. [Vdov06] ont décrit des architectures et des systèmes *P2P* qui soumettent des requêtes à des sources de donnés *RDF*. Ce travail montre comment dans des systèmes *P2P* tels que *Gnutella*[8] et *Freenet*[9] les ressources sont atteintes en utilisant une stratégie de routage générique (en passant d'un pair voisin à un autre pair voisin jusqu'à trouver la ressource). D'autres systèmes tels que *P-Grid*[10] et *CAN*[11] utilisent des index pour la recherche

[8] *Gnutella* : http://www.gnutella.com/
[9] *Freenet* : http://freenetproject.org/
[10] *P-Grid* : http://www.p-grid.org/
[11] *CAN* : http://www.can-cia.org/

d'information. Dans des systèmes comme *Edutella*[12] et *Piazza*[13], chaque pair ne connaît que les pairs avec lesquels il communique. Egalement, des systèmes tels que *Napster*[14] et *Gnutella* utilisent la réplication d'items à travers les pairs comme un mécanisme de tolérance aux fautes. Kokkinidis *et al.* [Kokk06] présentent différents projets basés sur l'approche *P2P* qui assignent des plans à chaque pair pour exécuter des tâches (individuelles ou collectives) telles que le routage et le traitement de requêtes à travers le réseau. Parmi les projets cités par ces auteurs, on trouve : *QueryFlow*[15], *Mutant Query Plans*[16], *AmbientDB*[17], *RDFPeers* [CaiF04], ou encore *Edutella* qui explore la conception et l'implémentation d'un schéma basé sur une infrastructure *P2P* pour le Web sémantique. Dans *Edutella*, le contenu de chaque pair est décrit par des schémas *RDF* et les super pairs sont responsables du routage de messages et de l'intégration/médiation des bases de données des pairs.

Compte tenu des caractéristiques d'un groupe de travail, des systèmes *P2P* tels que *KEx* [Boni06] sont utilisés, d'une part, pour la gestion de la connaissance d'un individu ou d'un groupe, et d'autre part, pour la coordination sémantique des pairs autonomes afin de simuler des *communautés de connaissances*[18] partageant des documents. La plate-forme *Xarop* [Llad06] est un système *P2P* sémantique composé d'une organisation virtuelle entre les agences de gouvernement et les compagnies de l'industrie touristique des Iles Baléares afin de promouvoir le tourisme dans cette région. Pour l'échange bibliographique entre chercheurs, Haase *et al.* [Haas06] proposent *Bibster* un système *P2P* qui permet aux utilisateurs *i)* de formuler une requête d'information à un seul pair, à un ensemble spécifique de pairs, ou au réseau entier de pairs ; *ii)* d'utiliser des recherches de mot-clés simples, mais aussi plus avancées (c'est-à-dire, par combinaison de valeurs spécifiques d'attributs) ; et *iii)* d'intégrer le résultat d'une requête dans une base de connaissances locale pour une utilisation future. Cette base de connaissances peut être partagée par d'autres pairs qui pourront (avec cette information) répondre aux requêtes qui leur ont été posées.

2.2.3 Processus de routage de requêtes

Afin de fournir aux utilisateurs nomades une information pertinente (c'est-à-dire « *une information correcte délivrée à l'endroit correct et au moment correct* »), une application exécutée sur des *DM* doit utiliser des mécanismes pour propager les requêtes de l'utilisateur vers les sources d'information « *les plus appropriées* » (stockées dans un ou plusieurs

[12] *Edutella* : http://edutella.jxta.org/
[13] *Piazza* : http://data.cs.washington.edu/p2p/piazza/
[14] *Napster* : http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/p4.html
[15] *QueryFlow* : http://www-db.in.tum.de/research/projects/OG/OnlineDemo/queryflow.shtml
[16] *Mutant Query Plans* : http://web.cecs.pdx.edu/~vpapad/mqp.html
[17] *AmbientDB* : http://homepages.cwi.nl/~boncz/ambientdb.html
[18] Une *communauté de connaissances* est un ensemble d'individus qui cherchent et fournissent des documents sur une base sémantique commune. Chaque individu (pair) fournit tous les services nécessaires à un autre individu pour créer et organiser sa propre connaissance locale (son autonomie) et définit des structures et des protocoles sociaux afin de chercher et partager des documents appartenants à d'autres individus.

dispositifs) qui peuvent répondre à ces requêtes en prenant en compte les *préférences de l'utilisateur*, les caractéristiques de son *DM*, sa localisation, *etc*. [Park04] [Sibe04]. C'est le principal but du processus de *routage de requêtes*, que nous décrivons dans cette section.

2.2.3.1 Définition

Xu *et al.* [XuLi99] définissent le *routage de requêtes* comme étant le problème général qui repose sur deux activités principales : d'une part, l'évaluation d'une requête en utilisant les sources d'information les plus pertinentes, et d'autre part, l'intégration des résultats retournés par les sources d'information. Afin d'accomplir ces deux activités, Xu *et al.* [XuLi99] identifient trois sous problèmes :

- La *sélection de sources d'information* qui nécessite d'analyser la requête de l'utilisateur afin de déterminer une ou plusieurs sources d'information aptes à y répondre. Afin de résoudre ce sous problème, le système doit connaître l'information gérée par chaque source ;
- L'*évaluation de la requête* de l'utilisateur par les sources d'information choisies pendant l'étape de sélection. En raison des éventuelles représentations hétérogènes des données dans les différentes sources d'information, la requête originale devra être traduite dans le vocabulaire des sources d'information interrogées ;
- La *fusion de résultats* faisant référence à l'intégration des résultats retournés par les différentes sources d'information.

Zhuge *et al.* [Zhug04] étendent la définition de Xu *et al.* en détaillant l'ensemble des tâches qui gèrent une requête : son traitement, sa traduction, son évaluation, la sélection de sources d'information qui peuvent y répondre, sa reformulation, son envoi et la compilation des résultats.

2.2.3.2 Routage de requêtes dans les systèmes P2P

Dans cette section, nous nous intéressons aux méthodes et aux algorithmes utilisés pour accomplir l'envoi d'une requête et la réception de ses résultats (provenant de diverses sources d'information), et la sélection des sources d'information appropriées vers lesquelles rediriger les requêtes.

a – Envoi d'une requête et réception de ses résultats

Le travail d'Agostini *et al*. [Agos04] détaille le processus d'envoi d'une requête par un pair. Après la formulation de la requête, un composant « *chercheur* » (« *seeker* ») vérifie les pairs connectés (« *actifs* ») et choisit, parmi eux, celui ou ceux à qui envoyer la requête (cf. section suivante). Ce composant « *chercheur* » utilise une stratégie de *confiance et réputation* (« *Trust and Reputation* » [Yang04]) afin de choisir le(s) pair(s) qui peu(ven)t répondre à la requête.

Agostini *et al*. montrent aussi le processus suivi lorsqu'un pair reçoit une requête. Ce pair essaie de résoudre la requête en utilisant sa connaissance. Chaque pair analyse les résultats provenant de ses pairs voisins, pour chacune des requêtes qu'il a émises, afin de les qualifier

comme appropriés ou non. La confiance des pairs qui ont répondu de façon appropriée est augmentée. La réponse est évaluée selon des critères établis tels que la vitesse de réponse, la qualité de la réponse, *etc*.

b – Sélection de sources d'information et redirection des requêtes

Plusieurs travaux ont été proposés afin de sélectionner de manière adéquate les sources d'information vers lesquelles rediriger les requêtes. Dans cette section, nous en présentons quelques uns en les classant en trois catégories : ceux qui regroupent des pairs pour répondre aux requêtes, ceux basés sur des filtres et la correspondance de données (« *matching* ») et ceux basés sur des stratégies de confiance et réputation (« *Trust and Reputation* »).

b.1 Stratégies regroupant des super pairs/pairs

Parmi les travaux qui regroupent des pairs capables de répondre aux requêtes, celui de Brunkhorst *et al.* [Brun03] permet de spécifier les pairs associés à un super-pair et l'information gérée par chacun dans le cadre de systèmes *P2P* basés sur des schémas. Les super-pairs composent un petit sous-ensemble de pairs, chacun ayant des responsabilités spécifiques et la capacité de se regrouper avec d'autres afin d'accomplir des tâches telles que le routage de requêtes, la médiation dans la résolution de conflits et le travail en groupe. Ces auteurs ont défini un ensemble d'index nécessaires pour le *routage de requêtes* entre les (super-) pairs du système tels que :

- Les *index de routage de super-pair/pair* contenant l'information sur les métadonnées (composées par des schémas et leurs propriétés) utilisées par chaque pair afin de décrire son contenu. Pendant l'enregistrement d'un pair auprès d'un super-pair, celui-ci fournit à son super-pair l'information sur son contenu et les schémas utilisés ;
- L'*index de schéma* contient l'identificateur du schéma ainsi que les identificateurs des pairs qui l'utilisent. Chaque schéma possède un identificateur unique (en utilisant un *URI*) et chaque pair peut supporter différents schémas ;
- L'*index de propriétés/ensembles de propriétés* permet aux pairs de décrire leurs contenus en sélectionnant des propriétés d'un ou plusieurs schémas. Les super-pairs utilisent un ensemble de propriétés afin de décrire leurs pairs ;
- L'*index pour le routage super-pair/super-pair* est défini afin de rediriger des requêtes entre les super-pairs.

Kokkinidis *et al.* [Kokk04] proposent de regrouper les pairs partageant les mêmes schémas d'information pour construire des plans afin de répondre aux requêtes d'une manière distribuée. Leurs contribution est un *middleware* nommé « *SQPair* » basé sur des *schémas actifs*[19] et qui est en charge de rediriger et de traiter des requêtes dans des systèmes *P2P*. Dans cette proposition,

[19] Un *schéma-actif* d'un pair est un sous-ensemble de schémas *RDF* demandés ou envoyés par « *broadcast* » à d'autres pairs afin qu'ils aient connaissance de l'information disponible dans les bases de données gérées par ce pair.

les tâches d'un pair et celles d'un super-pair sont bien définies. D'un côté, les super-pairs sont les responsables du routage de requêtes. Chaque nœud super-pair est aussi responsable de la création et de la gestion de son « *SON* » (« *Semantic Overlay Networks* »). Un *SON* est une manière de regrouper les pairs partageant les mêmes schémas d'information. Les pairs employant un ou plusieurs concepts de la même hiérarchie thématique sont sémantiquement associés et appartiennent au même *SON*. D'un autre côté, les sous-pairs sont responsables du traitement des requêtes. Dans *SQPair*, les requêtes sont formulées par les clients pairs en utilisant *RQL*[20], pour interroger les schémas *RDF* créés pour décrire les bases de données. Le traitement de requêtes en *SQPair* génère des plans de requêtes distribués compte tenu de l'information retournée par l'algorithme de routage. Ces plans de requêtes sont ensuite exécutés par les pairs les plus pertinents. Un pair peut facilement identifier des pairs pertinents car les pairs sont groupés par *SON*. Ce groupement réduit ou évite l'envoi de requêtes par « *broadcast* » à travers le réseau.

b.2 Filtres et mécanismes de correspondance de données (« *matching* »)

Koloniari *et al.* [Kolo04] définissent le *routage de requêtes* comme un mécanisme de détermination de la localisation des nœuds qui contiennent les documents nécessaires répondant aux requêtes. Cette localisation se base sur un mécanisme de correspondance de documents et des filtres. Les filtres sont des structures de données spécialisées qui regroupent l'information générale de larges collections de documents et qui redirigent les requêtes seulement vers les nœuds pouvant contenir l'information pertinente. Chaque nœud maintient deux types de filtres : le *filtre local* qui regroupe l'information générale des documents stockés localement dans le nœud, et les *filtres fusionnés* qui regroupent l'information générale des documents des nœuds voisins. Lorsqu'une requête atteint un nœud, ce nœud vérifie son *filtre local* et utilise les *filtres fusionnés* pour diriger la requête seulement vers les nœuds dont les filtres assurent la correspondance avec les requêtes. La similarité du contenu de deux nœuds est liée à la similarité de leurs filtres. L'opération de comparaison de filtres est plus efficace qu'une comparaison directe entre un ensemble de documents puisqu'un filtre d'un ensemble de documents est plus petit que les documents eux-mêmes. Le regroupement des nœuds repose sur la similarité de leurs contenus. Le but d'un tel regroupement est d'améliorer l'efficacité du routage de requêtes à travers la réduction du nombre de nœuds non pertinents traitant la requête. Lorsqu'un noeud *n* veut se joindre au système *P2P*, il en fait la demande (contenant son *filtre local*) à tous les nœuds racines. A la réception de cette demande, chaque nœud racine compare le *filtre local* reçu avec le *filtre fusionné* et répond à *n* en indiquant la mesure de la similarité de leurs filtres. Le nœud racine ayant la similarité la plus grande est le nœud « *gagnant* ». Le nœud *n* compare sa similarité avec le nœud « *gagnant* ». Si la similarité est plus grande qu'une valeur prédéfinie, *n* se joint à la hiérarchie de la racine gagnante, sinon *n* devient un nœud racine lui-même. Dans le premier cas, le noeud *n* demande à être rattaché à la racine gagnante. Cette dernière propage sa réponse à tous ses nœuds appartenant au sous-arbre de cette racine.

[20] http://139.91.183.30:9090/RDF/publications/www2002/www2002.html

Un autre système utilisant des filtres pour la gestion de requêtes est celui proposé par Idreos *et al.* [Idre04]. Appelé « *P2P-DIET* », ce système introduit un modèle de données simple (attribut-valeur) pour la description d'un réseau de ressources, et utilise *AWP*, un modèle de données pour spécifier des requêtes et des métadonnées de ressources textuelles. Idreos *et al.* proposent aussi « *BestFitTrie* », un algorithme de mémoire principale dont le rôle est de filtrer les documents nécessaires à l'élaboration d'une réponse à une requête. Cet algorithme utilise deux tableaux nommés « *Total* » et « *Count* ». Pour chaque requête dans la base de données, *Total* stocke le nombre de formules atomiques contenues dans cette requête. Quant à *Count*, ce tableau est utilisé pour compter le nombre de formules atomiques évaluées à vrai pour un document. Chaque élément de ces tableaux est initialisé à zéro au début de l'algorithme de filtrage (« *BestFitTrie* »). Si à la fin de l'algorithme, une entrée de la requête dans le tableau *Total* est égale à son entrée dans le tableau *Count*, alors le document répond à la requête.

Xu *et al.* [XuLi99] présentent quelques stratégies basées sur des techniques de « *clusters* » pour la sélection des sources d'information. Ces stratégies reposent sur deux étapes : d'un côté, la *construction de connaissances* à propos de sources d'information qui composent chaque cluster et sur l'information gérée par chacune. D'un autre côté, le *classement des sources d'information* est réalisé en fonction des réponses des sources d'un cluster. Un score de correspondance (« *matching* ») est calculé pour chaque cluster. Lorsqu'une requête correspond à un cluster, il est normal que plusieurs sources d'information de ce cluster répondent d'une manière pertinente à la requête. De plus, lorsqu'une requête correspond à un nombre significatif de clusters auxquels appartient la source d'information, on peut supposer que la source d'information est plus pertinente par rapport à la requête. Dès lors, la position de chaque source d'information dans le classement est déterminée tant par la somme des scores de correspondance entre les clusters auxquels la source appartient que par la taille des clusters.

b.3 Stratégies de confiance et réputation (« *Trust and Reputation* »)

Afin d'optimiser le processus de *routage de requêtes*, Agostini *et al.* [Agos04] proposent d'utiliser plusieurs métriques liées à la fiabilité des sources d'information, à leurs capacités à satisfaire les besoins d'information de l'utilisateur et à leur fiabilité pour délivrer l'information aux utilisateurs. Agostini *et al*. exposent une stratégie de *confiance et réputation* (« *Trust and Reputation* ») permettant de sélectionner les pairs vers lesquels rediriger les requêtes. Ces auteurs définissent la *réputation* comme les attentes à propos du comportement futur d'un pair en fonction de son comportement passé. Un système de réputation de pairs collecte, distribue et gère l'information sur le comportement passé des pairs.

La stratégie de *confiance et réputation* proposée par Agostini *et al.* [Agos04] repose sur le processus suivant : le *chercheur* (« *seeker* ») cherche à sélectionner les pairs capables de répondre à la requête *R* avec la plus haute probabilité par rapport à des critères établis (par exemple, la vitesse de réponse, la quantité d'information fournie, *etc*.). Afin de décider, le *chercheur* construit et gère une liste *<p1, p2,…pk>* de pairs dignes de confiance à qui soumettre la requête. La liste est ordonnée en utilisant un niveau de confiance décroissant. La stratégie suivie par le *chercheur* afin de répondre à une requête est la suivante : tout d'abord, il demande à *p1*, puis *p2*, et continue tant qu'il reçoit des réponses pertinentes. Il est important de remarquer

que la liste de pairs dignes de confiance peut évoluer. La confiance d'un pair est attribuée grâce à sa *réputation*.

Une stratégie similaire est celle présentée par Yang *et al.* [Yang04] qui utilisent comme devise : « *La performance passée est un bon indicateur pour la performance future* ». Cette idée est liée au concept de *Réputation*. Yang *et al.* introduisent le concept de *guide de routage* (« *Routing Guide* ») basé sur les résultats retournés par les requêtes précédemment traitées, et utilisé afin de déterminer les chemins de routage pour les requêtes suivantes. Dans le processus de récupération de documents en systèmes *P2P*, chaque nœud possède une collection de données (fichiers) partagée avec d'autres nœuds. Lorsqu'un utilisateur soumet une requête, son nœud devient la source de la requête (« *requestor* »). Une source peut envoyer des messages contenant la requête à plusieurs de ses voisins. Puis, un voisin recevant les messages contenant la requête se charge de la traiter en utilisant son information locale. Si le nœud peut répondre à la requête, il envoie alors les résultats à la source de la requête.

Röhm *et al.* [Rohm00] proposent un coordinateur simple et intelligent qui peut accomplir différentes tâches. Le coordinateur peut aussi diriger les requêtes vers le composant le moins chargé dans le cas de la réplication de données, décomposer et diriger des requêtes complexes en parallèle à plusieurs composants dans le cas de la partition ou de la réplication, et diriger et mettre en œuvre la réplication en parallèle. Röhm *et al.* [Rohm00] utilisent : le *routage de requêtes basé sur l'équilibre dans le nombre de requêtes* qui consiste à diriger une requête qui arrive au composant ayant le moins de requêtes actives, ou le *routage de requêtes basé sur l'affinité* qui consiste à grouper les requêtes accédant aux mêmes données et à diriger ce groupe aux composants gérant ces données. Afin de décider s'il y a affinité entre deux requêtes, il faut considérer et analyser les données accédées par les requêtes et les requêtes des sessions précédentes.

2.2.3.3 Traitement de requêtes

Brunkhorst *et al.* [Brun03] classent les mécanismes pour le *traitement de requêtes* en réseaux *P2P* en deux types compte tenu de la participation du client : *i)* Le *traitement de requêtes a complètement lieu sur le client*. C'est le cas du *traitement de requêtes en systèmes P2P purs et en systèmes P2P basés sur des schémas*. Pour les *systèmes P2P purs*, toutes les données requises doivent être expédiées au client et donc le réseau est inondé de requêtes. Les ressources (données requises) sont propagées à tous les voisins à travers le réseau. Pour les *systèmes P2P basés sur des schémas,* un index distribué guide sémantiquement la recherche des super-pairs appropriés pour la réponse des requêtes. Chaque fois qu'un nouveau participant se joint au système *P2P*, l'index est mis à jour par les nouveaux pairs avec les métadonnées des ressources. Toute l'information pertinente trouvée doit être expédiée au client qui établit des filtres sur l'information reçue. Néanmoins, les filtres définis par l'utilisateur ainsi que les opérateurs complexes peuvent seulement être appliqués après que les données aient été expédiées au client. *ii)* Le *traitement distribué de requêtes.* Dans ce cas, le client envoie la requête incluant les opérateurs définis par l'utilisateur au premier super-pair où les index locaux sont utilisés. La requête est décomposée en deux parts : l'opérateur à exécuter localement afin de combiner les résultats d'un côté, les parts à envoyer au (super) pair prochain, de l'autre.

Parmi les aspects à prendre en compte pour des applications relevant de l'*informatique ubiquitaire*, nous pouvons mentionner le besoin d'une infrastructure de communication permettant à différents pairs de partager un ensemble de ressources et des services afin d'accomplir une tâche individuelle ou collective [LiZo04a]. Cette infrastructure est fournie par les *systèmes P2P*. Un autre aspect important à considérer concerne les changements de localisation des pairs (pairs pouvant notamment représenter un utilisateur) et/ou de leurs dispositif d'accès. Ces changements peuvent mener aux changements des besoins d'information de tels pairs. Dans un environnement nomade, l'information requise par un pair peut évoluer en fonction de sa localisation. Par exemple, un utilisateur peut demander à un système la liste des restaurants situés dans la rue où il se trouve à un moment donné ou, quels sont les pairs les plus proches qui peuvent répondre à ses requêtes. Pour ces raisons, il est important de disposer de mécanismes permettant au système d'obtenir, de manière automatique, la localisation d'un pair, et de tenir compte du traitement de requêtes basées sur la localisation. Ces deux sujets sont l'objet de la section suivante.

2.3. Requêtes basées sur la localisation

Dans cette section, nous soulignons l'importance pour l'*informatique ubiquitaire* des requêtes basées sur la localisation et des mécanismes utilisés afin de capturer la localisation.

2.3.1 Requêtes basées sur la localisation

Cao *et al.* [CaoC04] définissent les *requêtes dépendantes de la localisation* comme celles dont le résultat est lié à une localisation qui n'est pas fournie explicitement dans la requête. Afin de construire la réponse à la requête, il faut tout d'abord connaître la localisation de l'utilisateur expéditeur de la requête (cf. section 2.3.2). En fonction de la localisation de l'utilisateur, les applications peuvent configurer leurs fonctionnalités (par exemple, pour une même activité d'un utilisateur, les applications peuvent exécuter des ensembles de fonctionnalités différents selon sa localisation) et les résultats des requêtes sensibles à cette localisation changent (par exemple, si l'utilisateur demande des restaurants situés dans la rue dans laquelle il se trouve, les résultats peuvent changer selon sa localisation).

Le travail de Thilliez *et al*. [Thil04] concerne également les « *requêtes basées sur la localisation* ». Ces auteurs distinguent les « *requêtes sensibles à la localisation* » (« *location-aware queries* ») et les « *requêtes dépendantes de la localisation* » (« *location dependent queries* »). Les premières sont exécutées indépendamment de la localisation de l'utilisateur (par exemple, *quels sont les sites touristiques les plus visités à Paris* ?). Les secondes sont évaluées en tenant compte de la localisation physique courante de l'utilisateur (par exemple, *quels sont les restaurants situés dans la rue où se trouve l'utilisateur* ?). Dans ce cas, les réponses visent à aider l'utilisateur à trouver facilement et précisément l'information localisée dont il a besoin.

La section suivante présente différents mécanismes pour obtenir la localisation de l'utilisateur. Nous considérons que cet aspect est essentiel pour l'adaptation de l'information, surtout si les requêtes de l'application sont *dépendantes de la localisation* et/ou si l'utilisateur considère explicitement la localisation comme un critère à prendre en compte pour adapter ou

chercher l'information. Il est important de noter que les changements de localisation de l'utilisateur peuvent entraîner la modification de ses activités dans le système.

2.3.2 Mécanismes de capture de la localisation

Les applications s'exécutant sur des *DM* peuvent obtenir la localisation de l'utilisateur ou de son dispositif d'accès de plusieurs manières : à travers une interface de saisie proposée à l'utilisateur ou à travers des mécanismes ou méthodes tels que ceux proposés par Nieto-Carvajal *et al.* [Niet04] comme le « *Signal Strength* », le « *SNMP* (*Simple Network Management Protocol*) » ou l'*Accès Local* à l'adresse *MAC*[21] du point d'accès. Indulska *et al.* [Indu03b] proposent l'utilisation de capteurs comme mécanismes d'obtention de la localisation de l'utilisateur ou de son dispositif d'accès. Ils distinguent les capteurs *physiques* des capteurs *virtuels*.

Les capteurs *physiques* fournissent l'information sur la position d'un dispositif physique. Les récepteurs *GPS*, des cameras et des lecteurs de code barres sont des exemples de ce type de capteurs. Ces capteurs fournissent de l'information sur la position ou la proximité du dispositif par rapport au système avec différents degrés de précision (c'est-à-dire, avec une précision de quelques centimètres, mètres ou centaines de mètres).

Les capteurs *virtuels* extraient de l'information de localisation de l'espace virtuel, c'est-à-dire, les applications logicielles, les systèmes d'exploitation et les réseaux. Ces capteurs peuvent suivre de près des événements d'applications (par exemple, des modifications dans les agendas d'un groupe d'utilisateurs), des événements du système d'exploitation (par exemple, la connexion/déconnexion d'un périphérique) ou des événements du réseau (par exemple, le changement d'adresse IP). L'information de localisation provenant de ces capteurs doit être combinée avec de l'information provenant d'autres sources d'information (par exemple, une base de données des localisations de dispositifs fixes) pour inférer l'information de localisation physique.

2.4. Conclusion

Lors de la conception d'une application s'exécutant sur des *Dispositifs Mobiles* (*DM*), il est nécessaire de tenir compte des contraintes de ce type de dispositif telles que des capacités de stockage et de traitement limitées, une hétérogénéité de la représentation des données, un manque de standards de spécification des tâches, un besoin de dispositifs embarqués pour la prise en compte des caractéristiques contextuelles de l'utilisateur, *etc*. D'ailleurs, l'utilisateur nomade souhaite disposer d'une information adaptée à ses caractéristiques et à celles de son dispositif d'accès. Cette information doit être fournie à l'utilisateur en contenu, forme, temps et lieu utiles. Ce sont les principes de base de l'*informatique ubiquitaire* qui est caractérisée par les

[21] Une **adresse MAC** (*Medium Access Control address*) est « un identifiant physique stocké dans une carte réseau ou une interface réseau similaire et utilisé pour attribuer mondialement une adresse unique au niveau de la couche de liaison (couche 2 du modèle *OSI*). Disponible sur : http://fr.wikipedia.org/wiki/Adresse_MAC.

dispositifs d'accès mobiles, l'infrastructure de communication largement distribuée et l'environnement hautement dynamique.

Dans un environnement nomade, il est nécessaire de prendre en compte que la connectivité du réseau ne peut pas être garantie à tout moment à tout endroit, et que le support de réseau pour l'*informatique ubiquitaire* doit permettre d'adapter le système aux possibles changements et d'anticiper les besoins et préférences de l'utilisateur.

L'*informatique ubiquitaire* ayant besoin d'architectures et de technologies de communication, nous nous sommes intéressés aux systèmes *P2P* qui reposent sur des architectures et des technologies permettant aux pairs de rester accessibles malgré les changements de localisation [LiZo04b]. Les pairs rejoignent ou quittent le système et changent constamment leurs contenus et rôles (tels que serveur, client, médiateur, *etc*.). Les systèmes *P2P* peuvent être « *non structurés* » (classifiés comme « *Purs* » ou « *Hybrides* ») ou « *structurés* ».

Compte tenu des caractéristiques des systèmes *P2P*, nous avons abordé le *routage de requêtes* dans ces systèmes qui désigne la propagation des requêtes de l'utilisateur vers les systèmes d'information capables de répondre à ces requêtes en fonction de critères tels que la localisation de l'utilisateur, ses préférences, les caractéristiques de son dispositif d'accès, *etc*. Nous avons présenté quelques-uns des travaux les plus représentatifs du domaine en les classant en trois catégories : ceux qui regroupent des pairs pour répondre aux requêtes, ceux basés sur des filtres et la correspondance de données (« *matching* »), et ceux basés sur des stratégies de *confiance et réputation* (« *Trust and Reputation* »).

Enfin, nous terminons ce chapitre avec la prise en compte de la localisation de l'utilisateur afin d'adapter les résultats de ses requêtes. Les requêtes qui prennent en compte la localisation sont classées en deux types [Thil04] : d'un côte, les « *requêtes sensibles à la localisation* » (« *location-aware queries* ») qui sont exécutées indépendamment de la localisation de l'utilisateur et, de l'autre, les « *requêtes dépendantes de la localisation* » (« *location dependent queries* ») qui sont évaluées en tenant compte de la localisation physique courante de l'utilisateur. Pour le deuxième type de requêtes, il est nécessaire d'obtenir des informations sur la localisation (par exemple, en demandant à l'utilisateur à travers une interface de saisie, ou à travers des capteurs) ; nous avons cité différents mécanismes d'acquisition de la localisation tels que des méthodes pour l'obtenir et des capteurs (physiques et logiques).

Les changements des caractéristiques contextuelles d'un utilisateur nomade (par exemple sa localisation, les conditions du réseau, les caractéristiques du *DM*) peuvent être gérés par des agents. Les propriétés des agents (par exemple, leurs connaissances, leurs capacités communicatives avec des utilisateurs ou d'autres agents, leur mobilité, leur autonomie, *etc*.) leur permettent de réagir, en sélectionnant les tâches les plus appropriées à exécuter sous les nouvelles caractéristiques. Une de ces tâches peut aussi être la représentation des caractéristiques d'un utilisateur à l'intérieur d'un système d'information.

Le chapitre suivant est consacré aux *Systèmes Multi-Agent*s (*SMA*) qui sont composés d'agents et qui peuvent être considérés comme des *Systèmes P2P*, dans lesquels chaque agent est un pair à part entière dans la mesure où il peut accomplir de manière autonome (en se basant

sur sa connaissance) ses tâches indépendamment du serveur et d'autres agents, et peut communiquer avec d'autres agents ou avec l'utilisateur à l'aide d'infrastructures de communication semblables à celles des systèmes *P2P*.

3. SYSTEMES MULTI-AGENTS ET REPRESENTATION DE CONNAISSANCES POUR LES ENVIRONNEMENTS NOMADES

La conception de *Systèmes d'Information basé sur le Web* (*SIW*) pour des *Dispositifs Mobiles* (*DM*) doit faire appel à des mécanismes et des architectures spécifiques afin de récupérer efficacement, stocker et délivrer des données. Les *SIW* doivent fournir à un utilisateur l'information qui lui est utile, en ayant recours à une recherche d'information qui tient compte des caractéristiques de l'utilisateur, et un affichage de l'information délivrée qui prend en compte les caractéristiques de son *DM*. Afin d'atteindre ce but, les *Systèmes Multi-Agents* (*SMA*) s'avèrent une approche pertinente. Selon Ramparany *et al.* [Ramp03], lorsque l'Internet est utilisé pour accéder et échanger l'information à travers des *DM* (qu'ils appellent « *dispositifs intelligents* »), l'intérêt des *SMA* réside dans deux aspects principaux : *i)* les agents sont utiles pour représenter les caractéristiques de l'utilisateur à l'intérieur du système, et *ii)* le *DM* peut travailler comme un « *dispositif coopératif* ». Les *SMA* sont définis par El Fallah-Seghrouchni *et al.* [ElFa04] comme un paradigme crédible pour la conception des systèmes coopératifs distribués basés sur la technologie agent. Un *SMA* est un outil utile pour modéliser un *SIW* – *SIW* qui devient aussi *SIWA*[22] – grâce aux propriétés des agents telles que la connaissance (définie, propre et acquise), la communication avec des utilisateurs ou d'autres agents, la mobilité, *etc*. Koch *et al.* [Koch04] classent des *SIWA* en prenant en compte :

[22] *SIWA* : *Système d'Information Web basé sur des Agents*. Dans notre travail, un *SIWA* est un *SIW* développé en utilisant une approche agent (et accédé par les utilisateurs à travers des *DM*). Un *SIWA* représente différentes applications telles qu'une guide touristique, le contrôle global de la circulation, etc. [Kuru04]

- La *granularité* liée au *nombre d'utilisateurs* qu'une application peut supporter : un (« *single user support* ») ou plusieurs utilisateurs (« *multi-user interaction supports* ») afin de coordonner des activités, d'organiser leurs agendas, leurs emplois du temps, *etc.* ;
- Le *rôle* lié à *l'activité principale* des agents dans une application : un agent peut *acquérir de l'information* (l'agent obtient l'information, l'enregistre pour son traitement futur, *etc.*), ou peut *l'analyser* et *l'afficher* (l'agent analyse l'information compilée et l'affiche), ou peut *proposer des actions* (l'agent analyse l'information compte tenu du contexte et des préférences de l'utilisateur afin de suggérer des actions), ou encore un agent peut *exécuter des actions* (l'agent possède les capacités pour accomplir les actions requises afin d'atteindre des buts) ;
- L'*autonomie* liée à la *façon de réagir* : un agent peut être *réactif* (ses actions sont déclenchées par l'utilisateur ou l'environnement) ou *proactif* (il répond à son environnement et il prend l'initiative en considérant le contexte, les préférences et le comportement de l'utilisateur).

Nous pouvons considérer que les *SMA* s'approchent des *Systèmes P2P* (cf. section 2.2) sur de nombreux points. Chaque agent est un pair à part entière dans la mesure où il peut accomplir ses tâches indépendamment du serveur et d'autres agents. Les *systèmes P2P* [Rohm00] sont caractérisés par : *i)* une communication directe entre pairs sans communication à travers un serveur spécifique ; *ii)* l'autonomie d'un pair pour accomplir les tâches qui lui sont assignées. Reposant sur une approche *P2P*, un *SIWA* doit représenter la connaissance requise par chaque agent afin d'accomplir les tâches associées à ses différents rôles (par exemple, *client*, *serveur*, *coordinateur*).

L'objectif de ce chapitre est de présenter les agents, leurs caractéristiques, les avantages de leur utilisation et leurs applications. Il est structuré de la manière suivante : la section 3.1 définit les caractéristiques principales des agents dans un *SMA*. Nous étudions particulièrement les caractéristiques des agents mobiles et des agents ubiquitaires. La section 3.2 présente des architectures basées sur des agents plus particulièrement celles qui permettent la recherche d'information dans différents *SIW* à travers des *DM*. Les activités d'interaction entre les agents (coordination, contrôle, coopération et négociation) nécessaires pour accomplir une tâche commune sont présentées dans la section 3.3. Les langages de description des interactions et des messages entre les agents sont aussi présentés. Finalement, dans la section 3.4, nous abordons la question des méthodes et des techniques pour la représentation et la gestion de la connaissance des agents.

3.1. Définition des Systèmes Multi-Agents et caractéristiques des agents

Carabelea *et al.* [Cara03] définissent un *SMA* comme « *une fédération d'agents logiciels*[23] *qui interagissent dans un environnement partagé coopérant et coordonnant leurs actions compte tenu de leurs buts et de leurs plans* ». D'après Korhonen *et al.* [Korh03], les agents utilisent une plate-forme qui fournit une infrastructure partagée et une interface pour l'envoi et la réception de messages. Ces messages peuvent être utilisés pour accéder à un *SIW* et pour échanger de l'information [*W3C*b05].

D'après Koch *et al.* [Koch04], le *paradigme agent* offre des méthodologies et des mécanismes pour la création d'applications distribuées, intelligentes, intégrées et coopératives. Ces auteurs, ainsi que Rahwan *et al.* [Rahw04], recommandent l'utilisation de ce paradigme pour les applications exécutées sur des *DM*. Un agent qui s'exécute sur des *DM* possède, en effet, des caractéristiques *proactives* et *adaptatives* qui lui permettent, dans un système ubiquitaire, de traiter l'information contextuelle de l'utilisateur et de son environnement afin de fournir à l'utilisateur une information adaptée. Un agent peut fournir des informations contextuelles (c'est-à-dire, temps de connexion, lieu et tâches sous exécution) au système auquel il accède. Dans un environnement ubiquitaire (cf. section 2.1), le contexte de l'utilisateur est dynamique puisque les utilisateurs peuvent se déplacer d'un endroit à un autre. Ces changements de localisation pourraient produire des changements dans les tâches et les besoins d'information de l'utilisateur. En conséquence, l'agent doit raisonner sur les buts de l'utilisateur et la manière dont ils peuvent être accomplis. Rahwan *et al.* [Rahw04] considèrent un *agent* comme une entité qui possède un comportement autonome et flexible où flexible signifie à la fois :

- *réactif*, pour percevoir l'environnement et répondre d'une manière opportune ;
- *proactif*, pour exhiber un comportement dirigé vers les buts à atteindre et pour prendre des initiatives grâce à sa connaissance (définie, acquise, inférée) ;
- *communicatif*, pour interagir avec d'autres agents ou humains.

Wooldridge *et al.* [Wool95] ont aussi mis en valeur le comportement flexible (*proactif*) et autonome des agents pour résoudre des problèmes, la richesse de l'interaction des agents (*caractéristiques de communication* avec d'autres agents ou avec l'utilisateur) en vue d'accomplir un objectif commun ou propre objectifs, et la complexité de la structure d'organisation d'un système d'agents (*sociabilité*). Un agent accomplit ses actions lorsqu'il se situe dans un environnement particulier (par exemple, informatique, physique, ubiquitaire) et il décide par lui-même du moment auquel doivent être accomplies ces actions.

Pour qu'un agent soit une entité réactive, proactive et communicative, Lopez y Lopez *et al.* [Lope02] le décrivent à travers une représentation composée d'un ensemble d'attributs (représentant ses caractéristiques permanentes), de l'ensemble des buts à accomplir, de

[23] D'après le [*W3C*b05], un *agent* est « *une pièce concrète logicielle ou matérielle qui envoie et reçoit des messages* »

l'ensemble de ses capacités d'exécution, et de l'ensemble de ses préférences. Cette représentation considère les agents comme des entités sociales capables d'éviter ou de résoudre des conflits, de concrétiser des accords, de réduire la complexité des tâches, et en général, d'accomplir des tâches communes. Les agents doivent être contrôlés et coordonnés pendant l'exécution de leurs tâches, et si toutes les tâches de l'agent aident l'utilisateur dans ses activités, c'est bien l'utilisateur qui doit toujours avoir le contrôle. Lopez y Lopez *et al.* [Lope02] soulignent le besoin d'établir des normes et des règles afin de contrôler le comportement de l'agent. Ils considèrent que les agents doivent être dotés de capacités à adopter de nouvelles normes. En matière de contrôle des activités des agents, Weib *et al.* [Weib03] proposent le concept de « *RNS* » (« *Roles, Norms, Sanctions* »), un schéma spécifiant les limites du comportement autonome de l'agent en termes de rôles à travers lesquels l'agent peut accomplir les buts et tâches assignés. Du point de vue de l'autonomie de l'agent, Wooldridge *et al.* [Wool95] considèrent qu'un agent est une entité « *stand-alone* » capable de représenter une personne ou une organisation. Bien qu'autonome l'agent est amené à évoluer au sein d'une communauté. A ces fins, l'*OMG*[24] établit des normes de comportement et d'exécution des agents.

L'exécution d'un agent peut être modifiée à cause de changements qui surviennent dans son environnement. L'environnement englobe des caractéristiques contextuelles, de la plate-forme d'exécution, des caractéristiques du système, des activités à accomplir, des protocoles de communication, *etc.*). D'après Hristova *et al.* [Hris03], un agent repose aussi sur un ensemble d'états mentaux, de règles de responsabilité (« *commitment rules* »), de « *perceptors* » et d'« *actuators* ». L'*état mental* d'un agent contient le modèle de son environnement actuel, stocké sous la forme de croyances. Les *règles de responsabilité* représentent le comportement d'un agent dans des situations où il faut adopter une responsabilité précise. Les *responsabilités* sont les résultats de la prise de décision de l'agent et représentent les actions qu'il doit assumer et exécuter. Ces actions sont exécutées par les « *actuators* » qui sont les unités fonctionnelles qu'un agent utilise pour modifier son environnement, tandis que les « *perceptors* » sont les unités fonctionnelles qu'un agent utilise pour construire (percevoir) le modèle de son environnement.

Certains agents peuvent être exécutés sur les *DM* ou migrer à travers le réseau afin de chercher l'information sur différents serveurs (ou *DM*) pour satisfaire les requêtes de l'utilisateur. C'est l'idée sous-jacente du concept d'*agent mobile* [Yang03]. D'après Lin *et al.* [LinL04], un *agent mobile* est un objet autonome actif qui peut migrer sur un réseau hétérogène et communiquer avec un gestionnaire d'*agents mobiles*, différents nœuds et d'autres *agents mobiles*. Les *agents mobiles* sont une alternative de programmation émergente pour le développement d'applications distribuées [Pogg04].

Kotz *et al.* [Kotz99] considèrent que chaque fois qu'un *agent mobile* migre d'un hôte à un autre, à travers le réseau, dans le temps et vers l'endroit de son choix, son état est sauvegardé dans l'hôte source et envoyé au nouvel hôte. La restauration de l'état de l'agent dans le nouvel

[24] *OMG* : Object Management Group.

hôte doit être garantie afin de permettre à l'agent de continuer dans le point d'exécution auquel il s'était arrêté avant de se déplacer. D'après Poggi *et al.* [Pogg04] un *agent mobile* n'est pas restreint à la plate-forme où il a commencé son exécution. Les systèmes d'*agents mobiles* se différencient des s*ystèmes de migration de processus* (« *Process-Migration Systems* ») en cela que les *agents mobiles* se déplacent lorsqu'ils le choisissent, typiquement en utilisant certaines primitives pré-établies (par exemple, « *jump* » ou « *go* »), tandis que dans un *système de migration de processus*, le système « *décide* » le moment et l'endroit pour déplacer le processus exécutant. D'après Kotz *et al*, les *agents mobiles* se différencient des « *applets* », en cela que les *agents mobiles* sont exécutés en intégralité sur un hôte tandis que les *applets* sont exécutés sur un client. Cependant, l'utilisation des *agents mobiles* pose des problèmes par rapport à la sécurité, à la portabilité, à la capacité d'évoluer (il n'y a pas de standardisation sur les environnements d'exécution) et à la performance (ils sont généralement écrits en langages interprétés). Ils n'ont pas de mécanismes ou d'applications pour arrêter leur exécution.

Dans des environnements ubiquitaires, les utilisateurs doivent pouvoir formuler des requêtes à tout moment et à tout endroit. Il est donc nécessaire de disposer de systèmes intelligents qui leur permettent d'accomplir cette consultation. C'est l'idée sous-jacente des *agents ubiquitaires* [Weny01]. De Carolis *et al.* [DeCa03] caractérisent les *agents ubiquitaires* en fonction de leur autonomie d'exécution et de leur capacité à communiquer avec d'autres agents afin de partager et d'échanger l'information pour accomplir des tâches individuelles ou collaboratives dans des environnements ubiquitaires. Si un *agent ubiquitaire* a besoin de migrer à travers le réseau afin d'accomplir ses tâches, cet agent devient un *agent ubiquitaire mobile*.

3.2. Architectures des Systèmes Multi-Agents

Durant la conception d'un *SMA*, il est nécessaire de définir les différents rôles que les agents peuvent jouer (par exemple, *client*, *serveur*, *modérateur*, *coordinateur*, *etc*.). Il est aussi nécessaire de préciser la manière dont un agent peut communiquer avec d'autres agents pour accomplir des tâches propres et/ou collectives. Panti *et al*. [Pant02] décrivent un *Système Multi-Agent Pair* basé sur le concept « *pair* ». Un *pair* représente une personne ou une organisation et possède un agent qui gère l'information qui lui est associée. Il possède aussi la capacité de gérer et de contrôler un « *workflow*[25] » simple.

Le but d'un agent pair est de communiquer et de partager des tâches et des ressources avec d'autres agents pairs dans un environnement dynamique sans l'aide d'un serveur explicite. De plus, un *agent pair* gère sa propre base de connaissances afin d'accomplir ses tâches. Cette connaissance est aussi utilisée pour représenter un client ou pour jouer le rôle d'un serveur selon le travail requis. En général, un *agent pair* peut jouer le rôle du serveur car il possède la connaissance pour le faire : il fournit un service d'annuaire (« *pages jaunes* ») pour chercher et

[25] Le terme de *workflow* fait ici référence à une séquence d'actions devant être accomplies dans le système. Il décrit les capacités qu'un agent doit posséder afin de gérer les tâches assignées, d'échanger l'information, d'interpréter les règles de comportement d'autres agents pairs, et finalement, de proposer les services de coordination et collaboration que chaque agent pourrait fournir au système [Pant02].

trouver ses pairs et il peut s'adapter lui-même aux changements du réseau. Ainsi, si des problèmes de réseau surviennent, un *agent pair* exécute ses tâches assignées sans la collaboration d'autres agents et les informe une fois que ces problèmes ont été résolus (en utilisant un mécanisme de *pages jaunes*).

L'architecture interne d'un *agent pair* permet de définir la stratégie qui répond aux requêtes de l'utilisateur. Elle est composée :

- Du *composant d'enveloppement* (« *wrapping* ») qui est une interface de communication entre des utilisateurs et le système, ou entre des agents.
- Du *composant de recherche/représentation* qui permet de stocker et d'exploiter une liste de tous les agents qu'il connaît (par exemple, les agents avec lesquels il a travaillé précédemment) et leurs services.
- Du *composant de reformulation/coordination* qui définit les tâches et les services que l'agent fournit au système afin de traiter une requête d'information. Les services et les tâches sont une partie de sa connaissance. De plus, l'agent connaît les tâches et les services d'autres agents qui peuvent l'aider à atteindre un but commun. Cette connaissance permet aux autres agents de définir le « *workflow* » afin d'assigner des responsabilités (il joue le rôle de coordinateur) pour accomplir ses tâches d'une manière collaborative/coopérative.
- Du *composant de génération des stratégies* qui décrit la stratégie utilisée pour résoudre un problème ou pour satisfaire une requête d'information.

Cependant, tout comme l'approche *P2P*, l'approche agent souffre du manque d'expressivité des langages de définition des données et des services, du traitement des problèmes propres des systèmes répartis (tels que l'hétérogénéité, la validité des données, entre autres), résultats de la distribution des données entre les *DM*, et de la non prise en compte des changements de caractéristiques contextuelles de l'utilisateur nomade.

Dans cette section, nous présentons trois architectures basées sur des agents pour la recherche d'information capables de tenir compte des caractéristiques de l'utilisateur et celles de son dispositif d'accès. Celles-ci cherchent l'information dans des *SIW* accédés à travers des *DM*.

KODAMA [Taka03] (***K****yushu University* ***O****pen and* ***D****istributed* ***A****utonomous* ***M****ulti-Agent* ***A****rchitecture*) est un système multi-agent conçu pour livrer de l'information publicitaire aux clients des centres commerciaux à Nagoya (Japon). Chaque client est muni d'un téléphone portable avec un transmetteur qui envoie des signaux aux récepteurs situés dans le centre commercial. Le récepteur le plus proche du téléphone portable du client active les agents du système de publicité (afin de chercher l'information sur les magasins qui sont proches du client) et envoie des messages au téléphone du client par le biais d'e-mails.

Son architecture multi-niveaux modélise des systèmes distribués basés sur des agents. *KODAMA* est composée de quatre niveaux :

- Le *niveau d'application* définit les tâches assignées à chaque agent à travers un ensemble de règles. Chaque règle a une priorité et une politique d'interprétation de messages envoyés par d'autres agents ;
- Le *niveau de communication* établit le mécanisme de communication entre agents. L'ensemble d'agents est structuré en une hiérarchie qui contient les relations de communication des agents. Chaque agent connaît les autres agents auxquels il peut demander des services ;
- Le *niveau d'infrastructure d'agent* connecte les deux niveaux précédents et simule une couche de réseau dans laquelle les agents sont des *nœuds* et leurs relations/communications sont des *arcs*. Il est en charge d'adapter l'échange de messages entre des agents ;
- Le *niveau de transport* fournit un service de transport pour envoyer les données entre les machines source et la cible à travers le réseau en utilisant des protocoles comme *TCP/IP*.

Dans *KODAMA*, le concept de *communauté* fait référence à un ensemble d'agents organisés d'une manière hiérarchique. Cette hiérarchie distingue un *agent portail* qui joue les rôles de *coordinateur* et de *contrôleur de profils* des autres *agents* dits *ordinaires*.

MIA [Beus00] (***M****obile* ***I****nformation* ***A****gent*) est dédiée à la recherche spécifique de données à travers des *DM* (tels que des PDA, des téléphones portables équipés d'un dispositif *GPS*, et des téléphones portables *WAP*). L'utilisation de *MIA* est basée sur deux principes : la localisation de l'utilisateur et l'accès ininterrompu et permanent à l'information. *MIA* est composé de :

- Un ensemble d'*agents mobiles* présents sur des *DM* permettant au système d'estimer leur localisation géographique (fournie par un dispositif *GPS* ou par l'utilisateur) et de communiquer avec un serveur situé sur l'Internet à travers des technologies sans fil ;
- Un *serveur* responsable de la communication entre l'application et les agents mobiles. Il utilise le protocole *http* afin de communiquer avec les agents mobiles et d'envoyer les résultats de la requête à travers des pages *HTML* et *WML* ;
- Un ensemble d'agents représentant le *modèle de Système d'Information.* Ces agents sont classés en quatre types : *i)* des agents utilisateurs (qui modélisent et contrôlent les activités de l'utilisateur), *ii)* des agents de localisation, *iii)* des agents araignées (agents intelligents du Web en charge de chercher l'information compte tenu des besoins de l'utilisateur), et *iv)* le « *Matchmaker* » (un agent portail ou intermédiaire entre le serveur et les autres agents, qui active les agents araignées pour commencer la recherche d'information).

Bien que *KODAMA* et *MIA* soient des architectures génériques pour modéliser des *SIWA*, elles ne spécifient pas la manière dont les agents communiquent, ni les services offerts par les agents à l'intérieur du *Système d'Information*.

CONSORTS [Kuru04] (*Architecture for* **CO***g***N***itive Re***SO***urce Management with physically-g****R****ounding agen****TS***) est une architecture d'agents ubiquitaires conçue pour le support

massif de *DM*. Elle détecte la localisation de l'utilisateur et définit son profil à travers un *Raisonneur Spatio-Temporel* (« *Spatio-Temporal Reasoner* ») qui gère le *Moteur d'Inférence Spatio-Temporel* (« *Spatio-Temporal Inference Engine* ») et la *Base de Données d'Information Temporelle* (« *Spatial Information Database* ») [Kuru04]. *CONSORTS* utilise la localisation et le profil de l'utilisateur afin d'adapter le contenu de l'information. Les agents conçus dans *CONSORTS* sont (cf. Figure 3.1) :

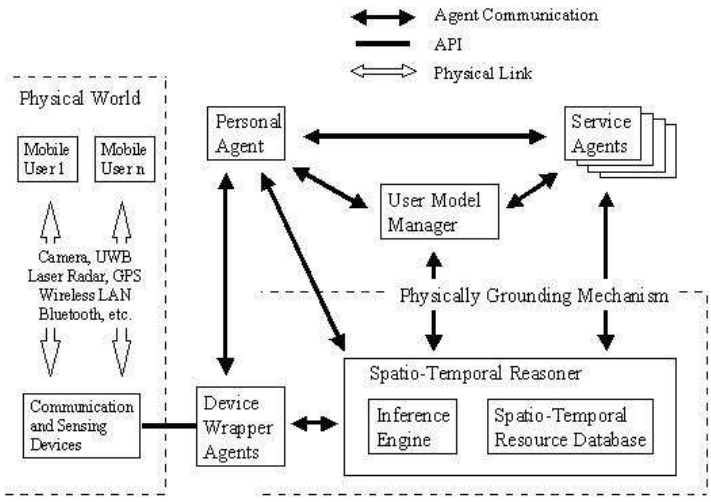


Figure 3.1. Architecture de CONSORTS (*D'après Kurumatani et al.* [*Kuru04*]).

- Le « *Dispositif Wrapper Agent* » qui est une interface de communication entre un *DM* et le système. Son objectif est de rendre la communication transparente pour les agents dans le *DM* avec le système sans considérer le type de *DM* utilisé (par exemple, *PDA*, téléphone portable) ;
- Le « *User Model Manager* » qui gère les profils des utilisateurs en utilisant un ensemble de règles d'inférence définies dans le *Spatio-Temporal Reasoner*. Ces règles sont définies selon les besoins fonctionnels du système. Le « *User Model Manager* » communique avec le *Spatio-Temporal Reasoner* afin d'obtenir la localisation de l'utilisateur (utilisée comme un critère pour définir le profil de l'utilisateur). Il y a trois caractéristiques prises en compte par le « *User Model Manager* » pour définir un profil de l'utilisateur : *i)* Les *intentions* qui sont l'ensemble de tâches qu'un utilisateur peut accomplir durant une période de temps (par exemple, si une photo est prise, l'utilisateur pourrait l'envoyer à un ami). *ii)* Les *préférences* qui sont l'ensemble de tâches que l'utilisateur voudrait accomplir durant une période de temps (par exemple, chaque fois que l'utilisateur prend une photo, l'utilisateur l'envoie à un ami). *iii)* Les *attributs* qui décrivent l'information personnelle de l'utilisateur (par exemple, l'utilisateur qui prend la photo est un photographe) ;

- Le « *Service Provider Agent* »[26] est le coordinateur du système et reste en communication permanente avec le « *User Model Manager* ». Il connaît les agents actifs dans le système, leurs services, contraintes et profils. Les profils sont définis selon les règles spécifiées dans le *Spatio-Temporal Reasoner* (par exemple, si l'utilisateur est dans la salle d'un musée, le *Service Provider Agent* fournit des informations sur les œuvres exposées et l'adapte aux contraintes de son *DM*) ;
- Le « *Service Requester Agent*[27] » est la représentation de chaque utilisateur dans le système. En interagissant avec le *Dispositif Wrapper Agent*, le *Service Requester Agent* permet aux utilisateurs de se connecter au système. Un *Service Requester Agent* connaît la localisation de l'utilisateur ; il est en charge d'interpréter les requêtes de l'utilisateur et de communiquer avec le *Service Agent* qui fournit l'information gérée par le système.

La troisième version de *CONSORTS* [Kuru04] introduit le « *support massif d'utilisateurs* » comme un service fourni à travers le concept de *coordination sociale* des agents. Cette coordination est une négociation automatique entre les agents *proxy* (*Service Requester Agents*) à l'aide d'un langage qui remplace les requêtes des utilisateurs. *CONSORTS* fournit quatre nouveaux services :

- Le *service d'adaptation* qui adapte les services du système compte tenu des caractéristiques de l'utilisateur (par exemple, localisation, profil, activités courantes) ;
- Le *service de combinaison* qui fournit des langages communs pour exprimer les services (c'est-à-dire une ontologie) implémentés dans des contextes différents ;
- Le *service de composition* qui permet de proposer des nouveaux services en composant des services déjà disponibles ;
- les *services* fournis par l'*agent sécurité* qui est le fournisseur d'un *framework* gérant les ressources logicielles pour les agents, contrôle leurs comportements dans le système et les élimine lorsqu'ils présentent des comportements anormaux. Un comportement anormal réfère à l'exécution des tâches en dehors ou en dépassant le rôle d'un agent.

CONSORTS [Kuru04] introduit aussi le « *Location-Aware Middle Agent* » qui aide le *Spatio-Temporal Reasoner* à localiser d'autres agents. Il est le médiateur entre le *Service Provider Agent* et le *Service Requester Agents* dans le *processus de coordination sociale*.

Le Tableau 1 récapitule les caractéristiques principales de ces architectures :

[26] Dans les versions précédentes de *CONSORTS*, le *Service Provider Agent* est nommé le *Service Agent* [Kuru04].

[27] Le *Service Requester Agent* est aussi nommé le *Personal Agent* [Kuru04].

	KODAMA	***MIA***	***CONSORTS***
Distribution de Données	plusieurs serveurs	plusieurs serveurs	serveur de contenu
Type de DM	*téléphone portable*	*PDA, téléphone portable, téléphone portable WAP*	*PDA, téléphone portable, ordinateur portable*
Types de données Multimédia	texte	texte	texte, images
Protocoles de Communication	*TCP/IP*	*http, WAP*	*http*
Définition de Règles pour les agents	oui	Inconnu	règles dans le *Spatio-Temporal Reasoner*
Mécanisme de Définition de Profil	*oui mais n'est pas défini de manière précise*	*préférences et localisation de l'utilisateur*	*préférences, intentions et attributs*
Mécanisme de détection de localisation	transmetteur dans le *DM* et récepteurs dans les lieux	GPS ou localisation entrée par l'utilisateur	capteur : camera ou LAN sans fil
Agent Coordinateur	*agent portail*	*serveur et Matchmaker*	*service provider agent*
Agent Portal	*agent portail*	*Matchmaker*	*Dispositif Wrapper Agent*
Type d'agents	agents mobiles, agents utilisateurs, agents de localisation, agents araignées, agent portail	mobile, utilisateur, localisation, agents araignées et *Matchmaker*	Spatio-Temporal, Utilisateur Model Manager, Service Provider, Service Requester, et Mobile

Tableau 1. Architectures pour modéliser SMA.

De l'étude des architectures *KODAMA*, *MIA* et *CONSORTS*, il ressort que trois niveaux communs composent l'architecture classique d'un *SIWA* :

- Un *niveau d'agents mobiles* qui est composé du *DM* et des *agents mobiles*. L'utilisateur nomade accède au système à travers son *DM* sur lequel les *agents mobiles* sont exécutés ;
- Un *niveau intermédiaire* qui offre des services (de connexion, de communication, *etc.*) afin de permettre l'interaction entre les deux autres niveaux ;
- Un *niveau de Système d'Information* qui représente les services (besoins fonctionnels) que le système fournit aux utilisateurs.

Dans la section suivante, nous soulignons les caractéristiques liées à la communication des agents afin d'accomplir ses tâches individuelles et pour travailler en groupe avec d'autres agents ou des utilisateurs.

3.3. Interaction entre agents

Wooldridge *et al.* [Wool95] et Rahwan *et al.* [Rahw04] soulignent la richesse de l'interaction des agents et montrent la nécessité pour les agents de disposer de mécanismes de communication leur permettant de résoudre d'une manière coopérative les problèmes, de coordonner et de synchroniser des actions, de résoudre des conflits, de participer à une négociation et d'envoyer de l'information. Cherchant à accomplir une tâche/un but commun de façon efficace, les agents exécutent quatre activités : la *Coopération*, la *Coordination*, le *Contrôle* de tâches et la *Négociation* (*CCCN*). Ces quatre activités présentées par Quintero *et al.* [Quin98] sont propres à l'interaction des agents, et peuvent impliquer de multiples échanges entre eux. Dans la suite, nous présentons la définition et les caractéristiques de chaque activité.

La première activité est la *coopération* pour l'exécution de tâches. Quintero *et al.* [Quin98] considèrent que dans un *SMA* co-existent deux types de tâches : les *locales* et les *globales*. Les tâches *locales* sont liées aux intérêts individuels de chaque agent, alors que les tâches *globales* sont liées aux intérêts globaux du système. Les tâches globales sont décomposées en sous-tâches (qui peuvent être locales ou globales selon la situation). Chaque sous-tâche est réalisée par un agent (compte tenu de ses capacités) en faisant l'hypothèse que l'intégration de la solution des sous-tâches amènera à la solution globale. La décomposition de la tâche globale ne garantit pas l'indépendance de chaque sous-tâche, raison pour laquelle il est nécessaire de compter sur des mécanismes de coopération qui permettent de partager des résultats partiels. Afin que les agents puissent coopérer de manière efficace, chaque agent doit posséder certaines capacités :

- posséder un modèle bien défini de son environnement qui lui permette de localiser les autres agents, de connaître la manière dont il peut communiquer avec eux, de connaître les tâches qu'ils peuvent réaliser ensemble, *etc.* ;
- intégrer l'information d'autres agents avec la sienne afin de créer des concepts globaux, c'est-à-dire, posséder une connaissance étendue ;
- interrompre un plan qui est en exécution afin d'aider d'autres agents et de coopérer avec eux si besoin.

La *coopération* dépend de la configuration organisationnelle du groupe d'agents. Si la structure est *centralisée*, les agents demandent la coopération à l'agent centralisé (exécuté sur la partie centrale du système). Si la structure est *hiérarchique*, la coopération peut être accomplie dans le même niveau ou à des niveaux différents. Si la structure est *horizontale*, la coopération est possible entre tous les agents.

La *coordination* est liée à la planification des actions pour la résolution de tâches. Les plans résultants permettent aux agents tant de connaître et de prédire le comportement d'autres agents

du système que d'échanger des résultats partiels amenant à la solution d'une tâche globale du système en évitant des actions redondantes[28].

La *coordination* peut suivre un modèle global (lorsque le *SMA* détermine et planifie globalement les actions des différents agents) ou un modèle individuel (lorsque le *SMA* donne une complète autonomie aux agents, c'est-à-dire lorsque chaque agent a deux responsabilités : décider ce qu'il doit faire mais aussi faire face aux conflits détectés impliquant d'autres agents). Il existe aussi deux types de coordination. D'une part, la *coordination orientée problèmes* dans laquelle les agents doivent coordonner les plans de réalisation d'actions afin d'empêcher des interblocages, des répétitions d'actions et la création d'inconsistances. D'autre part, la *coordination orientée coopération* dans laquelle les agents ne se coordonnent pas au niveau des plans, mais au niveau des actions.

Afin que les activités de *coopération* et de *coordination* soient réussies dans un *SMA*, une troisième activité s'impose pour que les agents puissent se mettre d'accord chaque fois qu'un agent défend ses propres intérêts. Cette activité, nommée *négociation*, doit amener à une situation bénéfique pour tous les agents du système. Le processus de *négociation* entraîne la modification ou la confirmation des croyances de chaque agent impliqué (concernant les autres agents ou son environnement). La négociation est vue comme un processus à six étapes : *i)* Définir le problème ; *ii)* Identifier les aspects de ce problème ; *iii)* Pondérer des critères ; *iv)* Générer des alternatives ; *v)* Evaluer les alternatives ; *vi)* Formuler la solution.

Le *contrôle de tâches* est le mécanisme qui sert de support à l'implémentation de mécanismes de *coordination* dans un *SMA*. Le contrôle est directement lié à quatre activités. Premièrement, déterminer les tâches les plus importantes à réaliser à un moment donné. Deuxièmement, déterminer le contexte (constitué notamment des résultats partiels d'autres agents) qui doit être utilisé dans la solution d'une tâche. Troisièmement, estimer le temps de génération de la résolution d'une tâche. Enfin, évaluer si un problème a été résolu.

Le *contrôle* peut être considéré comme *global* ou *local*. Le contrôle *global* correspond à une prise de décision qui se base sur des données obtenues et compilées à partir de l'information de tous les agents du système. Le contrôle *local* correspond à une prise de décision qui se base sur des données locales.

Les activités précédentes de *Coordination*, *Coopération*, *Contrôle* et *Négociation* (*CCCN*) des agents et leurs tâches dans un *SMA* ne sont possibles que si les agents possèdent des protocoles et des mécanismes de communication appropriés. Dans les sections suivantes, nous présentons les principaux langages utilisés pour l'interaction des agents.

3.3.1 Langage pour la modélisation de SMA

Afin de représenter les interactions entre agents Odell *et al.* [Odel01] et Poggi *et al.* [Pogg04] ont proposé un langage baptisé « *Agent Unified Modelling Language* » (*AUML*)[29], qui

[28] Le fait d'éviter des actions redondantes n'implique pas que dans certaines circonstances elles ne soient pas souhaitables.

est un ensemble d'extensions et d'idiomes d'*UML*[30]. *AUML* possède divers diagrammes permettant la modélisation des communications entre des agents :

- *Les templates et packages* sont des modèles paramétrés utilisés pour représenter des protocoles d'interaction d'agents (*AIP*, *Agent Interaction Protocols*[31]) et pour regrouper des agents selon un critère particulier (par exemple, leurs rôles, leurs fonctionnalités) ;
- *Les diagrammes d'activité et d'état* sont utilisés pour capturer tant la dynamique entre les agents que la dynamique à l'intérieur de chaque agent. Ils capturent le flux de traitement de données par les agents ;
- *Les diagrammes de collaboration et de séquence* sont utilisés pour capturer la dynamique entre les agents. D'un côté, les *diagrammes de collaboration* présentent les associations entre les agents, de l'autre, les *diagrammes de séquence* expriment la séquence chronologique des communications entre des agents ;
- *Les diagrammes de déploiement* représentent un graphe de *nœuds* connectés à travers des *associations*. Un *nœud* est un élément qui représente une ressource informatique possédant de la mémoire et une capacité de traitement d'information. Poggi *et al.* [Pogg04] utilisent les diagrammes de déploiement afin de représenter la mobilité des agents à travers une association entre deux nœuds. Un agent peut s'exécuter sur un nœud. Cette association utilise le stéréotype « *moves* », en spécifiant les déplacements de l'agent mobile d'un nœud (le nœud source) à un autre (le nœud destination) et le chemin (entre les deux nœuds) que l'agent mobile doit suivre. Une annotation indique l'objectif de ce déplacement. Ces diagrammes montrent des instances d'agents et intègrent d'autres stéréotypes représentant la mobilité tels que : « *home* » qui indique le nœud source de l'agent mobile, « *destination* » qui indique le nœud destination de l'agent mobile, « *visitor* » qui indique que l'agent mobile visite un nœud. Ces trois stéréotypes indiquent l'état de l'agent mobile. Il est nécessaire de préciser la direction du déplacement de l'agent. Lorsque le nœud destination manque, le nœud source devient la destination par défaut.

Odell *et al.* [Odel01] utilisent les *diagrammes de séquence AUML* afin de représenter les possibilités de concurrence à l'exécution d'agents. Certains cas de représentation sont illustrés ci-dessous :

- Tous les fils d'exécution FE_1 à FE_n sont envoyés de manière concurrente (Figure 3.2) :

[29] http://www.auml.org/

[30] *UML* : Unified Modelling Language. http://www.uml.org/

[31] Un *AIP* (*Agent Interaction Protocol*) permet de définir un patron de comportement commun à un ensemble d'agents. Un exemple d'*AIP* est le *FIPA Contract Net Protocol*. *http://www.fipa.org/specs/fipa00029/SC00029H.pdf*

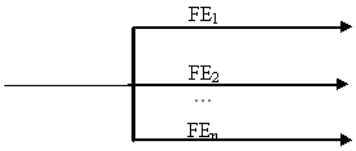


Figure 3.2. Diagramme de séquence de flux envoyés de manière concurrente.

– Le flux d'exécution est déterminé par une boîte de décision (cf. Figure 3.3). Cette décision concerne le nombre de flux (un ou plusieurs) qui seront envoyés. Si plus d'un flux est envoyé, la communication est concurrente :

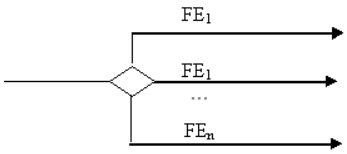


Figure 3.3. Diagramme de séquence de flux envoyés en tenant compte d'une condition.

– Le flux d'exécution est déterminé par une boîte de décision « *OR exclusive* » (Figure 3.4). Un seul flux d'exécution sera envoyé :

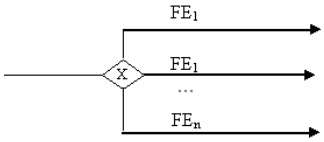


Figure 3.4. Diagramme de séquence de flux envoyés en tenant compte d'une condition exclusive n'envoyant qu'un flux d'exécution.

3.3.2 Langages de communication

Les agents communiquent entre eux à travers des messages. Pour cette raison, la *FIPA* (*Foundation for Intelligent Physical Agents*)[32] a défini des standards pour la structure des messages, leur représentation et les mécanismes pour leur transport. La *FIPA* a créé le *FIPA Agent Communication Language* (*FIPA ACL*)[33], un langage dont la spécification consiste en un ensemble de types de message (« *actes de communication* ») et en un ensemble de protocoles d'interaction de haut niveau.

La *FIPA*[34] a défini un « *acte de communication* » (« *communication act* ») comme une classe spéciale d'actions qui correspondent aux blocs les plus simples de dialogue entre les agents. Un *acte de communication* possède une signification bien définie, déclarative, indépendante du contenu d'un autre acte. L'idée de base des *actes de communication* est la

[32] http://www.fipa.org/specs/fipa00061/
[33] http://www.fipa.org/specs/fipa00003/OC00003A.html
[34] http://www.fipa.org/specs/fipa00037/SC00037J.html

théorie des « *actes de discours* » (« *speech act* ») proposée par Searle [Sear79]. Les *actes de communication* sont accomplis à travers l'envoi de messages d'un agent à un autre en utilisant les spécifications établies par la *FIPA*. Des exemples d'*actes de communication* sont : « Accept-proposal », « Agree », « Cancel », « Cfp », « Confirm », « Disconfirm », « Failure », « Inform », « Inform-if », « Inform-ref », « Not-understand », « Propose », « Query-if », « Query-ref », « Refuse », « Reject-proposal », « Request », « Request-when », « Request-whenever », « Subscribe », « Proxy », « Propagate ».

Un message *FIPA ACL* contient un ensemble de paramètres. Le seul paramètre obligatoire est la « *performative* » (c'est-à-dire, un *acte de communication*). Néanmoins, la plupart des messages doivent contenir des paramètres tels que l'expéditeur, le destinataire et le contenu. Les paramètres d'un message que la *FIPA* a spécifiés sont : « sender », « receiver », « reply-to », « content », « language », « encoding », « ontology », « protocol », « conversation-id », « reply-with », « in-reply-to », « reply-by ».

KQML[35] (acronyme de « *Knowledge Query and Manipulation Language* ») [Fini94] est un autre langage et protocole utilisé pour la création et le traitement de messages, ainsi que pour l'échange de l'information et de la connaissance entre agents. Il a été développé par l'« *ARPA Knowledge Sharing Effort* »[36] et peut être utilisé comme un langage de communication entre applications intelligentes qui partagent leurs connaissances afin de résoudre des problèmes coopératifs. *KQML* se focalise sur un ensemble extensible de « *performatives* » (telles que « ask-one », « ask-all », « stream-all », « advertise ») définissant les opérations permises entre les agents qui échangent connaissance et information. *KQML* fournit une architecture pour le partage de connaissances en utilisant des facilitateurs de communication (« *communication facilitators* ») qui coordonnent les interactions entre les agents. Un *facilitateur de communication* est un agent qui fournit plusieurs services de communication tels que le registre de services, l'envoi de messages, le routage de messages, les correspondances et la liaison entre les clients et les fournisseurs de services, la médiation et la traduction de services, *etc*.

3.4. Représentation de connaissances

Cette section porte sur la représentation et la gestion de connaissances des agents. Nous nous intéressons aux ontologies comme mécanisme de représentation de connaissances et aux règles comme mécanisme d'inférence de cette connaissance. Notre choix des ontologies et des règles parmi d'autres mécanismes de représentation (tels que la représentation de connaissances par objets, les logiques de descriptions, la logique des prédicats) et d'inférence de connaissances (par exemple la classification, l'héritage, les procédures) est motivé pour l'existence :

- de langages standard définis pour le Web qui expriment des règles et des ontologies. Par exemple *OWL* pour les ontologies, et *SWRL* et *RuleML* pour les règles ;

[35] http://www.csee.umbc.edu/kqml/

[36] « *ARPA Knowledge Sharing Effort* »[36] est un groupe de travail qui se focalise sur le développement de techniques et de méthodologies pour la construction de bases de connaissances (partageables et réutilisables).

- d'outils qui facilitent la définition d'ontologies (par exemple Protégé[37]) et la formulation et l'évaluation de règles (par exemple, le langage et le moteur d'inférence *JESS*[38]) ;
- de plusieurs bibliothèques qui permettent d'intégrer des règles et des ontologies aux plates-formes de développement d'agents. Par exemple, les classes[39] de la plate-forme *JADE*[40] qui permettent aux programmeurs de définir les ontologies des applications, et d'intégrer *JESS* pour définir et évaluer des règles ;
- de « *plug-ins* »[41] qui permettent de définir des ontologies et des règles et de les intégrer à une application. Par exemple, on peut définir une ontologie *Protégé* et la convertir en une ontologie *JADE* en utilisant le *plug-in* « *OntologyBeanGenerator* » de *Protégé* ;
- d'outils qui permettent d'intégrer des ontologies et des règles. Par exemple, le *plug-in* « *JadeJessProtege* » intègre *JADE, JESS* et *Protégé* en utilisant le *plug-in* « *JessTab*[42] » de *Protégé*.

3.4.1 Ontologies

Nous présentons des définitions d'une ontologie, ses caractéristiques et des langages pour sa représentation.

3.4.1.1 Définition

D'après Gruber [Grub93], une ontologie est « *une spécification explicite d'une conceptualisation* ». Dans la définition d'*ontologie* de Gruber, « *une spécification explicite* » signifie que les concepts et les relations d'un modèle abstrait reçoivent des noms et des définitions explicites. Une « *conceptualisation* » concerne un modèle abstrait qui représente la manière dont les personnes conçoivent les choses réelles dans le monde [Bucc05].

Plusieurs travaux précisent cette définition :

D'après Jean *et al.* [Jean06], une ontologie est considérée comme un nouveau type de *dictionnaire formel* et *consensuel* de catégories et de propriétés des entités d'un domaine ainsi que les relations qui existent entre elles et qui permettent de représenter des concepts d'une manière explicite. Le terme « *dictionnaire* » signifie que tout concept (entité) appartenant à l'ontologie peut être directement référencé. Chaque concept défini dans une ontologie possède un identificateur associé pouvant être référencé à partir de n'importe quel environnement, indépendamment d'un modèle d'ontologie particulier dans lequel ce concept a été défini. Le terme « *formel* » fait référence au fait qu'une ontologie est une conceptualisation basée sur une

[37] *Protégé* : http://protege.stanford.edu/ (cf. section 9.1.3)
[38] *JESS* : Moteur d'inférence, écrit en JAVA. http://www.jessrules.com/
[39] jade.content.onto.Ontology
[40] *JADE* : http://jade.tilab.com/
[41] http://jade.tilab.com/community-3rdpartysw.htm
[42] *JessTab* est un *plug-in* de *Protégé* qui fournit une console *JESS* permettant à l'utilisateur d'interagir avec *JESS* lors de l'exécution de *Protégé*.

théorie formelle permettant de vérifier un niveau de validité et d'accomplir un niveau de raisonnement sur les concepts et les individus définis par l'ontologie. Le terme « *consensuel* » fait référence au fait qu'une ontologie est une conceptualisation qui bénéficie de l'accord d'une communauté lors du développement d'une application particulière et commune.

Korhonen *et al.* [Korh03] considèrent qu'une ontologie peut être utilisée afin de partager de l'information, et Buccella *et al.* [Bucc05] montrent la manière dont une ontologie fournit d'un côté, le nom et la description des entités d'un domaine spécifique en utilisant des prédicats représentant des relations entre ces entités, et de l'autre, un vocabulaire pour représenter et communiquer la connaissance du domaine à travers un ensemble de relations contenant les termes du vocabulaire à un niveau conceptuel. D'après Pan *et al.* [PanC03], une ontologie fait aussi référence à un ensemble de contraintes du domaine en restreignant l'interprétation de son vocabulaire.

Du point de vue des agents, une *ontologie* définit les termes utilisés dans des messages codifiés en utilisant un *Langage de Communication d'Agents* (cf. section 3.3.2) et fournit donc un mécanisme permettant aux agents d'avoir une compréhension partagée de la sémantique des termes utilisés dans ces messages. Les ontologies étant partagées entre l'agent émetteur et le (les) agent(s) récepteur(s) font que les messages peuvent être interprétés correctement.

D'après Jean *et al.* [Jean06] et Buccella *et al.* [Bucc05], les ontologies peuvent être aussi utilisées pour la *recherche* et l'*intégration de données.* Pour *la recherche*, car une ontologie fournit un accès aux données qui référencent les concepts qu'elle définit. Pour l'*intégration de données* puisque les ontologies permettent, d'une part de capturer et d'identifier des concepts provenant de plusieurs sources de données éventuellement hétérogènes d'une manière automatique, formelle et unique et, d'autre part, de supporter la gestion de la validité des données et l'identification de données inconsistantes.

En ce qui concerne l'intégration de données, Buccella *et al.* [Bucc05] montrent que les ontologies sont utiles à la résolution du problème d'hétérogénéité des sources d'information. Ces auteurs classent l'hétérogénéité en quatre types : *i)* s*tructurelle* (impliquant différents modèles de données), *ii) syntaxique* (impliquant différents langages et représentations de données), *iii) systémique* (impliquant différents configurations matérielles et systèmes d'exploitation), et *iv) sémantique* (impliquant différents concepts et leurs interprétations). L'hétérogénéité sémantique tient compte de concepts sémantiquement *équivalents*, *liés* et *non-liés*. Des concepts sont *sémantiquement équivalents*, si un modèle utilise différents termes afin de faire référence au même concept. Par exemple, des synonymes ou des propriétés modélisées différemment par divers systèmes (le concept de « *longueur* » est modélisé en *mètres* dans un système et en *miles* dans un autre). Des concepts sont *sémantiquement non-liés*, si le même terme peut être utilisé par divers systèmes afin de dénoter des concepts complètement différents. Par exemple, dans un système, « *Java* » correspond à un langage de programmation, dans un autre système « *Java* » correspond à une danse. Finalement, des concepts sont *sémantiquement liés* si différentes classifications sont possibles pour un même terme. Par exemple, dans un système, une « *personne* » peut être classée comme « *homme* » ou « *femme* », alors que dans un autre système, une personne est classée comme « *professeur* » ou « *étudiant* ». Afin de répondre

à une requête dont les résultats proviennent de plusieurs sources d'information, un *SIW* peut définir une ontologie permettant de tenir compte de leur hétérogénéité.

Afin de stocker et de mettre à jour des ontologies, les registres d'ontologies (« *ontology repository* ») ont été créés. Un exemple de tel registre est celui présenté par Pan *et al.* [PanC03] qui définissent une méta-ontologie pour modéliser la structure des ontologies stockées dans un registre. L'architecture du registre est basée sur *http* et *RDF* en utilisant *REST*[43]. Pour *REST*, une ressource (qui possède un *URN*[44] comme identificateur) est considérée comme l'unité fondamentale d'information. Les ressources sont des notions abstraites qui ne sont pas directement transférées à travers le réseau. En revanche, c'est la représentation de l'état d'une ressource qui est transférée. Le registre stocke les multiples représentations de chaque ontologie en utilisant différents types de média. Le travail de Pan *et al.* [PanC03] propose aussi un serveur d'ontologies fournissant le support pour créer et éditer des ontologies, publier et récupérer des ontologies, enregistrer des métadonnées sur les ontologies et les relations entre elles, naviguer de manière interactive à travers la structure d'une ontologie et, disposer de mécanismes d'inférence afin de vérifier la validité des ontologies.

3.4.1.2 Langage de représentation

Le *W3C* a proposé *OWL*[45] (acronyme de « *Ontology Web Language* ») un standard pour la définition, la publication et le partage d'ontologies visant le traitement sémantique du contenu de l'information disponible sur Web. *OWL* peut être utilisé explicitement afin de représenter un ensemble de concepts et leurs relations. De plus, il possède un pouvoir expressif plus fort que celui de *XML*, *RDF*, et *RDF-S*. *OWL* ajoute un vocabulaire afin de décrire des propriétés, des classes et leurs relations. Ce vocabulaire offre un ensemble riche de types de propriétés avec leurs caractéristiques (par exemple, la symétrie), permet de préciser la nature des relations entre classes (par exemple, la disjonction) et leur cardinalité (par exemple, « exactement un »), et donne la possibilité d'énumérer des classes. *OWL* se décline en trois niveaux de détail correspondant aux trois sous langages *OWL Lite*, *OWL DL*, et *OWL Full*.

– *OWL Lite* fournit aux utilisateurs une classification hiérarchique et des contraintes simples. Il supporte toutes les contraintes d'*OWL DL* mais interdit l'utilisation de propriétes telles que : « *owl:oneOf* », « *owl:unionOf* », « *owl:complementOf* », « *owl:hasValue* », « *owl:disjointWith* », « *owl:DataRange* ». L'idée sous-jacente d'*OWL Lite* est de fournir un sous-ensemble minimal utile des caractéristiques d'*OWL* qui sont relativement simples, ainsi qu'une hiérarchie de base : des sous-classes et des contraintes sur les propriétés. Les implémentations qui supportent le vocabulaire

[43] *REST* : « *REpresentational State Transfer* » est un style architectural pour les systèmes hypermédias distribués qui vise à minimiser la latence et les communications du réseau. En même temps, *REST* maximise l'indépendance et le passage à l'échelle lors de l'implémentation des composants [PanC03].

[44] *URN* : Uniform Ressource Name

[45] *OWL* : http://www.w3.org/TR/owl-features/

d'*OWL Lite* font qu'un système *OWL* peut interagir avec des modèles *RDFS*, des bases de données ;

– *OWL DL* supporte un niveau d'expressivité supérieur à celui d'*OWL-Lite* et inclut tout le vocabulaire d'*OWL*. Ce vocabulaire ne peut être utilisé que sous certaines contraintes (par exemple, alors qu'une classe est sous-classe de plusieurs classes, une classe ne peut pas être une instance d'une autre classe). *OWL DL* est nommé de cette manière en raison de sa correspondance avec les *logiques de description* (« *description logics* ») [Baad03], un domaine de recherche qui fournit à un constructeur d'ontologies ou à l'utilisateur un support pour le raisonnement ;

– *OWL Full* consiste en la combinaison de l'expressivité d'*OWL*, la flexibilité et les caractéristiques de modélisation de *RDF* (par exemple, dans *OWL Full* une classe peut être traitée simultanément comme une collection d'individus et comme un individu). Cependant, l'utilisation d'*OWL Full* signifie la perte de quelques garanties qu'*OWL DL* et *OWL Lite* peuvent fournir pour les systèmes de raisonnement. *OWL Full* permet à une ontologie d'augmenter la signification du vocabulaire (*RDF* ou *OWL*) pre-défini. Il est improbable que tout système de raisonnement soit capable de garantir la complétude des raisonnements pour toutes les caractéristiques d'*OWL Full*. *OWL Full* contient tout le langage *OWL* et fournit une utilisation libre et sans contraintes de *RDF*. Dans *OWL Full* la ressource « `owl:Class` » est équivalente à « `rdfs:Class` ». Ce n'est pas le cas dans *OWL DL* et *OWL Lite*, où « `owl:Class` » est une sous-classe propre de « `rdfs:Class` » (ce qui signifie que toutes les classes *RDF* ne sont pas des classes *OWL* dans *OWL DL* et *OWL Lite*). *OWL Full* permet aussi aux classes d'être traitées comme individus. Dans *OWL Full* les valeurs des données sont considérées comme une part du domaine des individus. En effet, dans *OWL Full* l'univers des individus englobe toutes les ressources (« `owl:Thing` » est équivalent à « `rdfs:Resource` »). Cela signifie que les propriétés des objets et les propriétés des types de données ne sont pas disjointes. Dans *OWL Full* « `owl:ObjectProperty` » est équivalent à « `rdf:Property` ». La conséquence est que les propriétés des types de données sont effectivement une sous-classe des propriétés des objets.

OWL représente un domaine particulier d'une manière structurée en utilisant des classes et des propriétés, organisées en taxonomies. D'autres travaux comme celui de Golbreich [Golb04] utilisent des *logiques de description*[46] qui permettent de définir un domaine particulier en termes d'individus, concepts et rôles.

Dans la section suivante, nous présentons le formalisme des règles sur lesquelles se greffe un mécanisme de gestion et d'inférence de connaissances et nous présentons des standards pour leur définition et représentation.

[46] Les *Logiques de Description* (« *Description Logics, DL* ») sont une famille de formalismes de représentation de connaissances qui utilise le paradigme de modélisation orientée objet afin de décrire un ensemble de concepts correspondant à un domaine particulier.

3.4.2 Règles

Les règles sont un mécanisme d'inférence de connaissances [Ross03] [Golb04]. Dans le domaine des ontologies, les règles s'appliquent sur des faits qui représentent les concepts d'un domaine. Une règle est une contrainte explicite sur des comportements, fournissant un support à ces faits et à la conduite des activités. Dans la suite, nous présentons des travaux qui utilisent des règles pour la gestion des connaissances. Nous montrons la manière dont ces travaux combinent des ontologies et des règles exprimées à l'aide de standards du Web Sémantique.

3.4.2.1 Classement

Mckenzie *et al.* [Mcke04] classent les règles en cinq types :

- Les *règles de dérivation* qui calculent une valeur dérivée afin de fournir des vues sur les données stockées dans des sources d'information ;
- Les *règles de réécriture* qui sont utilisées pour l'optimisation de requêtes et pour le remplacement d'une expression par une autre expression équivalente mais impliquant généralement moins d'accès aux sources d'information ;
- Les *règles evénement-condition-action* (*Event-Condition-Action, ECA*) qui peuvent être définies comme : « lorsqu'un *événement* se produit, si la *condition* de la règle est satisfaite, alors, déclencher l'action de la règle » [47] (par exemple, si un utilisateur ferme sa session, l'agent qui s'exécute sur son *DM* aura comme état « *mort* ») ;
- Les *contraintes quantifiées* dans lesquelles toutes les variables libres sont associées à un quantificateur universel ou existentiel. Ces contraintes sont très appropriées pour exprimer des sémantiques de domaine spécifique pour les collections de données stockées ;
- L'*utilisation de quantificateurs mixtes en contraintes qualifiées* puisqu'une contrainte quantifiée peut faire cesser l'existence d'un fait ou peut faire que quelques relations deviennent fausses.

3.4.2.2 Langages de règles

Différents langages ont été établis afin de définir des règles. Nous mentionnons les langages les plus utilisés pour les applications sur le Web :

- *RuleML*[48] (« *Rule Markup Language* ») a été développé par la « *Rule Markup Initiative* », une initiative internationale pour la standardisation des règles d'inférence. Ce langage est basé sur *XML* [Golb04] ;

[47] http://scott.univ-lehavre.fr/~sadeg/recherche/publications/articles/tsi99/TSI_1999.pdf
[48] *RuleML* : http://www.ruleml.org/

- *SWRL*[49] acronyme de « *Semantic Web Rule Language* » est un langage qui combine *OWL* et *RuleML*. Ce nouveau langage étend la syntaxe abstraite et les axiomes existants d'*OWL* tels que *subClassOf, equivalentClass, disjointWith,* entre autres ;
- *JESS* est un moteur de règles et un environnement de « *scripts* » utilisé pour la construction d'applications *JAVA* qui possèdent la capacité de « *raisonner* » en utilisant des connaissances exprimées sous forme de règles déclaratives.

Dans la suite, nous présentons des travaux qui combinent des règles et des ontologies dans un seul langage en utilisant des langages définis pour le Web. Concernant les extensions de *SWRL*, on peut citer les travaux permettant d'intégrer des règles *SWRL* et des ontologies (exprimées en *OWL*) en utilisant *Racer*[50] et *JESS* [Golb04], d'inclure des contraintes quantifiées dans *SWRL* [Mcke04], et de traduire des règles *SWRL* en *SWRLx, RuleML, JESS* et *pseudo-Prolog* [Math04]. Nous présentons ci-dessous chacun de ces travaux :

Le travail de Golbreich [Golb04] propose un *plug-in* de *Protégé* nommé « *SWRLJessTab* » qui permet de faire des inférences en utilisant *Racer* et *JESS* afin de raisonner avec des règles *SWRL* et des ontologies représentées en *OWL*. Golbreich considère la correspondance entre les rôles *SWRL* et les règles *JESS* et entre les individus *OWL* et les faits *JESS*. Par exemple, pour chaque règle *SWRL* définie en *Protégé*, l'ensemble d'atomes de son « *antécédent* » correspond à une clause *IF* (nommé « *Left-Hand-Side* ou *LHS* ») et l'ensemble d'atomes de son « *conséquent* » correspond à la clause *THEN* (nommé « *Right-Hand-Side* ou *RHS* »). D'après Friedman-Hill [Frie06], une règle est assimilée à une instruction « `if... then` » en langage procédural. Pour certains moteurs d'inférence (tels que *JESS*), une règle n'est pas analysée de manière procédurale. Dans un langage procédural, une instruction « `if... then` » est exécutée en suivant l'ordre dans lequel elle apparaît dans le programme. Pendant l'exécution d'un moteur d'inférence, une règle est exécutée seulement lorsque sa clause « `if` » (la *LHS*) est satisfaite. De ce fait, l'exécution de règles de moteurs d'inférence est moins déterministe que dans un programme procédural typique.

Le travail de Mckenzie *et al.* [Mcke04] présente une extension de *SWRL*, nommée *CIF* (« *Constraint Interchange Format* ») qui consiste en une représentation de contraintes quantifiées. Cette représentation gère les quantificateurs existentiels et universels d'une manière explicite et compatible avec *OWL* et *RDFS*. *CIF* est un langage expressif de contrainte quantifiée basé sur la logique de premier ordre, et dérivé des langages de contraintes/requêtes de Colan [Bass95]/Daplex [Ship81].

Matheus [Math04] propose une syntaxe de présentation de *XML*, nommée *SWRLp* qui facilite la lecture et l'édition de règles *SWRL*. Il propose aussi des « *scripts* » pour la traduction de *SWRL* en *SWRLx, RuleML, JESS* et *pseudo-Prolog*. L'information pertinente sur les prédicats et les termes a été déplacée aux éléments noms et valeurs d'attributs afin de faire des règles plus

[49] *SWRL* : (http://www.w3.org/Submission/SWRL/)
[50] *Racer* : Raisonneur d'OWL. http://www.sts.tu-harburg.de/~r.f.moeller/racer/

concises et lisibles en suivant la syntaxe *pseudo-Prolog* tout en tenant compte des contraintes du langage *XML*.

Les travaux de Mei *et al.* [MeiL04] et Laera *et al.* [Laer04] concernent l'intégration d'*OWL* avec des règles *ASP*[51] et *OWLRuleML*[52], respectivement. D'un côté, le travail de Mei *et al.* [MeiL04] présente une extension d'*OWL* (similaire à *ORL*[53]) avec des règles *ASP* qui considèrent la négation classique (pour exprimer de l'information négative : $\neg p$ qui signifie « *nous croyons que p est faux* ») et la négation par défaut (pour modéliser de l'information incomplète et des exceptions : *not p* signifie « *il n'y pas de raison de croire en p* »). D'un autre côté, Laera *et al.* [Laer04] proposent *SweetProlog*, un système pour traduire des règles Web en *Prolog* permettant l'intégration d'ontologies et de règles sur le Web Sémantique. Cette intégration est accomplie à travers d'une part, une traduction d'ontologies *OWL* décrites en utilisant des *logiques de description* et d'autre part, une traduction des règles exprimées en *OWLRuleML*, en *Prolog*. Les avantages d'utiliser des *logiques de description* d'*OWL* (qui spécifient une méta-ontologie pour des règles *OWLRuleML*) sont essentiellement les suivants : les classes *OWL* peuvent être utilisées comme prédicats de règles, les règles et les axiomes d'ontologie peuvent être librement mêlés, et les ontologies et les règles peuvent être analysées grammaticalement de la même manière. *OWLRuleML* est utilisé car il fournit un support pour des références d'*URI*, la négation forte et la négation faible et le traitement de conflits en utilisant des priorités.

SweetProlog possède cinq fonctions principales : *i)* la traduction des ontologies *OWL* et des règles *OWLRuleML* en triplets *RDF* (un triplet *RDF* consiste en un sujet, un prédicat et un objet) ; *ii)* La traduction d'assertions *OWL* en *Prolog* (les triplets *RDF* représentent des concepts et des instances *OWL* qui sont traduits en prédicats *Prolog*) ; *iii)* La traduction de règles *OWLRuleML* en règles *CLP*[54] ; *iv)* La transformation de règles *CLP* en règles *Prolog* ; *v)* L'interrogation des sorties de *CLP* (les prédicats traduits sont passés par un moteur *JIProlog* capable d'inférer de nouvelles connaissances).

3.5. Conclusion

Ce chapitre a été consacré aux agents, entités intelligentes qui possèdent des comportements proactifs et autonomes afin de résoudre des problèmes, et des capacités d'interaction grâce auxquelles chaque agent peut communiquer avec d'autres agents ou avec l'utilisateur afin d'accomplir un objectif commun ou propre. Les agents appartiennent à une organisation appelée généralement *Système Multi-Agents* (*SMA*). Dans une telle organisation, un

[51] *ASP* : *Answer Set Programming*.

[52] Les règles *OWLRuleML* font référence aux ontologies *OWL*. Les noms de prédicats dans ces règles sont des *URI* qui associent les classes (prédicats unaire) et les propriétés (prédicats binaires) dans une ontologie *OWL*. Les assertions sur les instances de classes et de propriétés sont vues comme faits.

[53] *ORL* : *OWL Rules Language*. C'est une extension d'*OWL* avec des règles de clauses de Horn.

[54] *CLP* : « *Courteous Logic Programs* ». *CLP* fournit une méthode afin de résoudre des conflits entre règles en utilisant information partiellement prioritaire pour garantir un ensemble de conclusions unique et consistent (answer-set).

agent accomplit ses actions lorsqu'il se situe dans un environnement particulier et il décide lui-même du moment auquel doivent être accomplies les actions.

Nous avons étudié des architectures telles que *KODAMA* [Taka03], *MIA* [Beus00] et *CONSORTS* [Kuru04] qui utilisent des agents et définissent des composants et des caractéristiques générales de communication pour qu'un *SIW* soit accessible à travers des *DM*. De cette étude, nous avons identifié trois niveaux qui composent l'architecture classique d'un *SIWA* :

- Un *niveau d'agents mobiles* qui est composé du *DM* et des *agents mobiles*. L'utilisateur accède au système à travers son *DM* sur lequel les *agents mobiles* sont exécutés ;
- Un *niveau intermédiaire* qui offre des services (de connexion, de communication, *etc*.) afin de permettre l'interaction entre les deux autres niveaux ;
- Un *niveau Système d'Information* qui représente les services (besoins fonctionnels) que le système fournit aux utilisateurs.

Afin d'exploiter les caractéristiques des agents, nous nous sommes intéressés aux travaux concernant :

- La Coopération, la Coordination, le Contrôle de tâches et la Négociation entre agents qui permettent l'interaction des agents, la résolution coopérative des problèmes, la coordination et la synchronisation des actions, la résolution de conflits, la participation dans une négociation et l'envoi de l'information ;
- Les mécanismes et les langages de communication pour l'échange de messages ;
- La représentation de la mobilité des agents, l'exécution concurrente et l'échange de messages ;
- La représentation et l'inférence de connaissances.

Plusieurs sorties fonctionnelles et techniques doivent être considérées dans la modélisation d'un *SI*. Le système doit d'une part, répondre aux requêtes de l'utilisateur, et d'autre part, adapter l'information en considérant les préférences de l'utilisateur, les caractéristiques de son *DM* et celles du *contexte d'utilisation* : sa localisation, les activités qu'il veut accomplir dans le système, le moment de connexion, *etc*.

Les données sur le *contexte d'utilisation* permettent notamment (mais pas seulement) d'obtenir une description des conditions (temporelles, spatiales, matérielles, *etc*.) dans lesquelles l'utilisateur accède au *SI*. Les données sur les *préférences de l'utilisateur* déterminent la configuration du système souhaitée par l'utilisateur (les fonctionnalités, le contenu, l'affichage, *etc*.). L'adaptation de l'information vise donc à délivrer à un utilisateur une information en adéquation avec son profil, et de fait, avec les préférences que celui-ci inclut. Cette adaptation change la fonctionnalité, l'interface, la teneur en information ou l'aspect d'un système pour augmenter sa pertinence personnelle. La sélection des préférences qui peuvent être appliquées lors de la session en cours d'un utilisateur repose sur le *contexte d'utilisation*.

Le chapitre suivant est dédié à l'*adaptation* de l'information et nous présentons quelques travaux basés sur des agents qui adaptent l'information aux utilisateurs compte tenu du *profil de l'utilisateur* (basé sur ses préférences) et des caractéristiques contenues dans le *contexte d'utilisation*.

4. ADAPTATION DANS DES ENVIRONNEMENTS NOMADES

L'objectif visé par l'adaptation lors de l'utilisation de *Systèmes d'Information basés sur le Web* (*SIW*) depuis des *Dispositifs Mobiles* (*DM*) est de fournir à l'utilisateur nomade une information qui corresponde au mieux à son *contexte d'utilisation* courant (c'est-à-dire, celui de la session en cours). Par « *contexte d'utilisation* », nous faisons référence à un ensemble de données qui permet de caractériser la situation dans laquelle s'inscrit l'interaction entre l'utilisateur et le système. Un *contexte d'utilisation* repose sur une représentation de divers éléments tels que les activités effectivement menées par l'utilisateur, les caractéristiques du dispositif utilisé, la localisation ou encore le moment auquel a lieu la connexion. L'un des défis posés aux concepteurs de *SIW* accessibles par des *DM* est que le *contexte d'utilisation* d'un utilisateur nomade, est très variable et en constante évolution [Tami06] [Bouc06]. Les mécanismes de représentation et d'exploitation de ce *contexte d'utilisation* en vue de l'adaptation doivent être capables de supporter une telle évolutivité. Les travaux menés récemment au sein de notre équipe de recherche par [Kirs06] constituent une proposition en ce sens. Ce travail permet de décrire des *contextes d'utilisation potentiels* auxquels le système sait réagir, c'est-à-dire sait s'adapter. L'approche consiste à appliquer un algorithme de comparaison visant à confronter la description du contexte courant d'utilisation à celles de ces contextes potentiels. Une fois que l'appariement réalisé (lorsque le contexte courant est assimilé à un contexte connu), le système applique les actions préconisées pour adapter sa réponse à ce contexte.

Afin de chercher l'information dans des *SIW* et de l'adapter au *contexte d'utilisation*, Zemirli *et al.* [Zemi05] modélisent l'utilisateur comme une composante du processus de recherche d'information qui consiste à « *délivrer l'information pertinente en fonction des caractères spécifiques de l'utilisateur, adapter les résultats de recherche aux attentes de*

l'utilisateur et idéalement, les précéder ». Pour Bouzeghoub *et al.* [Bouz05], la pertinence de l'information (mais aussi des services) est notamment évaluée en fonction d'un ensemble de préférences spécifiques à un utilisateur ou à une communauté. Ces préférences sont généralement utilisées afin de décrire les centres d'intérêt de l'utilisateur, le niveau de qualité de données souhaité par l'utilisateur ou la manière de présenter ces données. Un tel ensemble de préférences entre généralement dans la constitution du (voire constitue pleinement) le *profil utilisateur* [Koch02] [Bouz05] [Tami06] [Zemi05]. L'adaptation d'un processus d'accès à l'information vise donc à délivrer à un utilisateur une information en adéquation avec son profil, et de fait, avec ses préférences. Cette adaptation change la fonctionnalité, l'interface, la teneur en information ou l'aspect d'un système pour augmenter sa pertinence du point de vue personnel de l'utilisation. Ceci requiert de disposer d'outils permettant de représenter les profils des utilisateurs et de mécanismes pour exploiter ces représentations en vue de l'adaptation des *SIW*. Dans notre travail, nous nous intéressons plus particulièrement aux outils de représentation des préférences et aux mécanismes dédiés. D'autres éléments qui peuvent entrer dans la composition d'un profil de l'utilisateur et ainsi le caractériser – par exemple, un niveau de connaissances ou son âge –, ne sont pas étudiés ici ni exploités dans notre approche. D'autres caractéristiques sont aussi prises en compte à des fins d'adaptation telles que la localisation de l'utilisateur, les activités qu'il veut accomplir dans le système, le moment de connexion, *etc*. Toutes ces caractéristiques composent le *contexte d'utilisation*.

Ce chapitre est structuré de la manière suivante : la section 4.1 présente la définition de l'adaptation de l'information, ses types et quelques modèles utilisés afin d'adapter l'information. La section 4.2 est consacrée à la représentation du profil de l'utilisateur. Des travaux sur la gestion des préférences de l'utilisateur sont présentés dans la section 4.3. La définition et la gestion du contexte d'utilisation sont présentées dans la section 4.4. Finalement, nous présentons dans la section 4.5 des travaux basés sur des agents qui adaptent l'information à l'utilisateur compte tenu de ses caractéristiques et de celles de son dispositif d'accès.

4.1. Adaptation des Systèmes d'Information

Quelques uns des aspects les plus importants à prendre en compte lors de la conception des *SIW* sont :

- L'augmentation de l'efficacité de l'utilisateur mesurée en temps dépensé dans la recherche d'information ou en quantités d'informations utilisées réellement par l'utilisateur [Dolo02] ;
- L'*adaptation* de l'information en prenant en compte les caractéristiques de l'utilisateur, ses besoins, buts, tâches, connaissances ou préférences [Yude05] ;
- L'adaptation de l'information en tenant compte des caractéristiques du dispositif d'accès.

Comme le soulignent Cannataro *et al.* [Cann01], les besoins des applications ont changé à cause de l'existence de *différents types d'utilisateur* (à cause de la démocratisation de l'accès au Web, l'hétérogénéité des utilisateurs dû à différents intérêts, buts, conditions sociales, *etc*.) mais

aussi de *différents types de dispositifs d'accès et de réseaux* (d'un côté, les dispositifs d'accès possèdent différents logiciels et interfaces, et d'un autre côté, les réseaux possèdent différentes propriétés telles que la bande passante, la latence, *etc*.). Dans des environnements nomades, il est aussi nécessaire de prendre en compte les conditions dans lesquelles un utilisateur évolue. C'est pourquoi, Cannataro *et al.* proposent de considérer l'adaptation à travers trois dimensions : le comportement de l'utilisateur (par exemple, ses activités dans le système), la technologie utilisée (par exemple, le réseau, le dispositif de l'utilisateur), et finalement, l'environnement externe (par exemple, le temps de connexion, la localisation, *etc*.).

D'après Murray *et al.* [Murr00], un utilisateur qui accède aux *Systèmes d'Information* (*SI*) et navigue à travers l'information délivrée par les *SI*, peut être exposé à une série de problèmes qui pourraient être résolus en utilisant des techniques d'adaptation. Parmi ces problèmes, Murray *et al.* citent la désorientation en référence aux utilisateurs qui ne savent pas où ils sont, ni où ils sont déjà allés ; la surcharge cognitive en référence aux utilisateurs qui sont débordés par les nombreuses options d'information et de navigation disponibles ; et l'absence de personnalisation du contenu à l'utilisateur (qui fait, par exemple, que l'utilisateur s'ennuie si le contenu est trop facile ou se sature si le contenu est trop difficile à comprendre).

Dans cette section, nous présentons différents types d'adaptation et les modèles proposés pour adapter l'information.

4.1.1 Types d'adaptation

D'après Schwinger *et al.* [Schw05], l'adaptation peut être classée selon le genre d'adaptation (les changements qui doivent être faits), le sujet d'adaptation (ce qui est à changer) ou le processus d'adaptation (caractérisé par la manière dont l'adaptation est accomplie). En ce qui concerne le *genre*, l'adaptation dispose d'opérations telles que les filtres de contenu, l'addition de liens, *etc*. Le *sujet* concerne le niveau de l'application Web qui est affecté par l'adaptation (par exemple, le contenu, la structure de navigation ou la présentation). Chaque niveau contient des éléments qui peuvent être adaptés tels que des pages, liens, structures d'accès, types de média, *etc*. Le *processus* comprend un nombre de tâches qui peuvent être adaptées pendant l'exécution d'une application. L'adaptation de l'information peut être accomplie, compte tenu des caractéristiques de la session en cours ou de l'historique des sessions de l'utilisateur.

D'après Brusilovsky [Brus00] et Henze *et al.* [Henz02], deux aspects (relevant de la notion de *sujet* chez Schwinger *et al.*) sont susceptibles d'être adaptés à un utilisateur dans un système : *i)* le *contenu* (adaptation réalisée en tenant compte des caractéristiques de l'utilisateur, aussi nommée « *adaptation au niveau du contenu* » ou « *présentation adaptative* ») ; *ii)* les possibilités offertes à l'utilisateur de *naviguer* dans les documents hypermédias d'une manière personnalisée. Cette dernière adaptation est aussi nommée « *adaptation au niveau des liens* » ou « *support adaptatif de navigation* ».

Dans les sous-sections suivantes, nous présentons les caractéristiques de ces adaptations.

4.1.1.1 Adaptation au niveau du contenu

L'objectif de l'*adaptation au niveau du contenu* est d'adapter le contenu d'une page hypermédia aux préférences [Bail02b], buts, connaissances et autres informations stockées dans le *modèle de l'utilisateur* [Brus00]).

Koch *et al*. [Koch02] proposent un *modèle de l'utilisateur* composé des attributs de l'utilisateur et de leurs valeurs pertinentes pour l'application adaptative. Ce modèle repose sur la connaissance et l'expérience de l'utilisateur liées au domaine d'application, ainsi que sur des caractéristiques générales de l'utilisateur (par exemple, préférences, tâches, *etc*.). Yudelson *et al*. [Yude05] définissent un *modèle de l'utilisateur* comme un ensemble de caractéristiques (personnelles, démographiques, du contexte), de données (in)dépendantes du domaine d'application, *etc*.

Afin de former un modèle global de l'utilisateur, Raza *et al*. [Raza02] proposent une ontologie collectant un ensemble de cas passés, similaires à la situation courante de l'utilisateur. Pour adapter le contenu de l'information, Koch *et al*. [Koch02] présentent un modèle orienté objet dont le *modèle d'utilisateur* est spécifié en utilisant *UML* et un mécanisme d'adaptation (basé sur des aspects du système tels que les besoins, les préférences ou la connaissance de l'utilisateur) en utilisant *OCL*[55].

Afin d'adapter le contenu de l'information, plusieurs travaux ont été proposés. Un premier exemple est le travail de Raza *et al.* [Raza02] qui utilise une technique d'adaptation compositionnelle sélectionnant et collectant les items d'information les plus pertinents provenant de cas passés afin de les regrouper. Un deuxième exemple est le système proposé par Bailey *et al.* [Bail02b], nommé « *Queries in Context, QuIC* », qui traite les contenus des pages visitées par l'utilisateur et augmente la page en cours avec des liens associés à ces contenus. Pour contrôler cette augmentation (nommée « *Link Augmentation* »), le contexte est défini par deux aspects : les *intérêts de l'utilisateur* et le *contexte d'un document hypermédia.* Ce dernier aspect tient compte, d'un côté de ses caractéristiques telles que son contenu, son format (html, pdf, gif, *etc*.), son but, la date de création, le serveur dans lequel il est stocké et, de l'autre, de sa relation avec d'autres documents.

4.1.1.2 Adaptation au niveau de la navigation

L'objectif de l'*adaptation au niveau de la navigation* est d'aider les utilisateurs à trouver leur chemin dans l'hyperespace en adaptant la présentation et les fonctionnalités des liens à leurs buts, leurs connaissances et autres caractéristiques [Brus00]. Ce support se focalise sur des aspects de la navigation sur des hyperliens tels que la génération, l'apparence, le placement spatial et la fonctionnalité. [Bail02a]).

[55] *OCL* : « *Object Contraint Language* ». Ce langage permet de spécifier tant les invariantes pour les éléments du modèle que les pré-conditions et post-conditions des opérateurs en décrivant la fonctionnalité adaptive.

Plusieurs auteurs présentent différentes approches pour l'*adaptation au niveau de la navigation* : Dolog *et al.* [Dolo02] utilisent des diagrammes d'état, d'un côté, pour modéliser des possibles chemins à travers l'hypertexte et, de l'autre, pour déterminer les caractéristiques structurelles et de comportement du *modèle de l'utilisateur*. Medina-Medina *et al.* [Medi03] ont proposé les *routes guidées*[56] (« *Guided Routes* ») personnalisées afin de faciliter la navigation. Pour la construction d'une *route guidée*, le système utilise un arbre de navigation basé sur les préférences de l'utilisateur. Dans cet arbre, chaque *nœud* représente un état de connaissance que l'utilisateur peut atteindre et chaque *arc* représente la visite d'un item. La route est obtenue en accomplissant une recherche dans l'arbre de navigation. Le nœud de départ représente l'état de connaissance de l'utilisateur et le nœud final représente un état de connaissance qui fait la correspondance avec le but de l'utilisateur. Bailey *et al.* [Bail02a] proposent un processus (nommé « *Link augmentation* ») qui consiste à insérer dynamiquement des liens (« *links* ») dans une page web existante. Son avantage est de laisser intact tant la structure de navigation sous-jacente de la page Web que les hyperliens originaux. Cependant, il y a toujours le risque de saturation de liens (chaque mot pouvant devenir un lien).

Dans la section suivante, nous présentons les *systèmes hypermédias adaptatifs* qui adaptent l'information en prenant en compte les deux niveaux présentés ci-dessus compte tenu des caractéristiques de l'utilisateur telles que ses besoins, buts, tâches, connaissances, préférences, *etc*.

4.1.2 Systèmes Hypermédia Adaptatifs

Brusilovsky [Brus96] définit un *Système Hypermédia Adaptatif* comme « *tout système hypertexte et hypermédia reflétant des caractéristiques de l'utilisateur dans son propre modèle d'utilisateur, modèle utile pour adapter plusieurs aspects du système à l'utilisateur* ». Selon Brusilovsky [Brus00], les *systèmes hypermédia adaptatifs* construisent un modèle de buts, de préférences et de connaissances de chaque utilisateur individuel et utilisent ce modèle à travers l'interaction avec l'utilisateur afin d'adapter l'information à ses besoins. Bailey *et al.* [Bail02b] affirment que dans les *systèmes hypermédias adaptatifs*, l'adaptation est assurée par les données accumulées, obtenues en observant les caractéristiques du contexte de l'utilisateur telles que sa connaissance, ses tâches, ses attitudes, ses intérêts, *etc*. En résumé, Medina-Medina *et al.* [Medi03] caractérisent un *système hypermédia adaptatif* comme « un *système hypermédia*[57] possédant un *modèle de l'utilisateur* nécessaire pour l'*adaptation* du système ».

Pour les *systèmes hypermédias adaptatifs*, trois modèles sont pris en compte : le *modèle de domaine d'application*, le *modèle de l'utilisateur* et le *modèle d'adaptation*. D'après Cannataro *et al.* [Cann01], Seefelder de Assis *et al.* [Seef04] ou encore Koch *et al*. [Koch02], le *modèle de domaine d'application* décrit les contenus hypermédias de base et leur organisation, le *modèle*

[56] Une *route guidée* (« *Guided Route* ») est un ensemble d'items et un ordre pour les visiter.

[57] Les *systèmes hypermédias* ont deux éléments : des *nœuds* et des *liens*. Les *nœuds* sont les unités d'information fournies par le système et peuvent être au format texte, image, audio, vidéo, etc. Les *liens* connectent les nœuds et permettent à l'utilisateur de naviguer à travers le système [Medi03].

de l'utilisateur décrit ses caractéristiques, ses préférences et ses expectatives, et le *modèle d'adaptation* repose sur des techniques pour adapter des présentations par rapport au comportement de l'utilisateur et aux buts du fournisseur de contenu.

Pour Raad *et al.* [Raad02], un *système hypermédia adaptatif* est composé de trois sous-systèmes : *i)* le *sous-système hypermédia traditionnel* contenant le modèle de navigation, le modèle d'interface (spécifiant les entités qui seront affichées à l'utilisateur final) et les ressources multimédias ; *ii)* le *sous-système adaptatif* contenant le modèle de l'utilisateur, le modèle de réseau sémantique (constitué d'une organisation de concepts et de relations entre ces concepts), l'analyseur des événements (qui organise les événements provenant de l'utilisateur) et l'entité de spécification du comportement adaptatif (basée sur des différentes catégories d'utilisateurs) ; *iii)* le *modèle de tâches* contenant les règles de stratégies associées avec l'entité de spécification du comportement adaptatif, les règles de navigation associées au modèle de navigation et les règles d'interface associées au modèle d'interface.

Parmi les applications des *systèmes hypermédias adaptatifs*, Murray *et al.* [Murr00] présentent une application du « *Metalinks project*[58] » fournissant un *framework* et des outils pour la création d'« *hyper-books* ». Un « *hyper-book* » est un document hypermédia, tel qu'un site web éducatif ou un *CD-ROM*, contenant un ensemble cohérent de sujets organisés et écrits à des fins éducatifs. Il peut être construit afin d'accentuer différents buts et des niveaux de compétence. L'approche de Conlan *et al.* [Conl02] est utilisée pour la composition d'un cours personnalisé reposant sur trois modèles indépendants : le *modèle pédagogique* (qui adapte le contenu d'un cours à la connaissance de l'étudiant), les *modèles de contenu* et *de l'utilisateur*. Cette approche a été définie pour la composition dynamique et la livraison d'apprentissage personnalisé en utilisant des objets d'apprentissage réutilisables. Henze [Henz03] présente une relation entre le Web sémantique et les systèmes hypermédias éducatifs adaptatifs. Pour cet auteur, les *systèmes hypermédias adaptatifs* mêlent des idées des systèmes hypermédia et des systèmes de tutorat intelligents (« *intelligent tutoring systems* »), et sont capables de personnaliser l'accès à l'information. L'environnement hypermédia adaptatif de Garlatti *et al.* [Garl03] se base sur la construction de documents virtuels qui consistent en un ensemble de fragments d'information, des ontologies et un moteur de composition capable de sélectionner les fragments d'information les plus appropriés et de les organiser compte tenu de la structure d'un document général en adaptant plusieurs aspects du document délivré à l'utilisateur.

Néanmoins, les travaux sur les *systèmes hypermédias adaptatifs* ne spécifient pas la manière d'adapter l'information aux caractéristiques des environnements nomades telles que la *localisation* (pouvant modifier les besoins d'information de l'utilisateur), et les caractéristiques du *dispositif d'accès mobile* (contraignant l'affichage de l'information).

Dans la section suivante, nous présentons les principaux modèles utilisés pour l'adaptation de l'information, pris en compte pendant la conception de systèmes d'information.

[58] *MetaLinks project* : http://www.tommurray.us/index.asp

4.1.3 Modèles utilisés pour l'adaptation

Selon Garlatti *et al.* [Garl03], lors de la conception d'un système d'information, il est nécessaire de définir quatre ontologies pour adapter l'information : *i)* une ontologie de *domaine* représentant les contenus. Cette ontologie définit un vocabulaire partagé qui est utilisé tant dans le schéma de métadonnées pour la description du contenu des données que dans le modèle de l'utilisateur pour la description de connaissance; *ii)* une ontologie de *métadonnées* décrivant la structure de fragments d'information ; *iii)* une ontologie de l'*utilisateur* définissant les stéréotypes et les caractéristiques individuelles ; *iv)* une ontologie de *documents* permettant de représenter les compétences de l'utilisateur et sa connaissance.

Afin de faciliter la navigation de l'utilisateur dans les systèmes d'information, Medina-Medina *et al.* [Medi03] ont proposé quatre sous-systèmes : *i)* un système de *mémorisation* permettant de construire le modèle conceptuel du système (lié à l'ontologie de domaine) ; *ii)* un système de *présentation* cherchant à choisir différentes présentations d'un modèle conceptuel ; *iii)* un système d'*apprentissage* déterminant quoi, à qui, comment et quand adapter l'information (lié à l'ontologie de documents et de l'utilisateur) ; et *iv)* un système de *navigation* définissant un ordre pour récupérer les pièces d'information d'un modèle.

A propos des *SIW* qui adaptent l'information aux utilisateurs en se focalisant sur l'accès aux données et à la présentation, dans notre équipe, Villanova [Vill02] a proposé *KIWIS*, un environnement dédié à la génération automatique de *SIW*. *KIWIS* repose sur cinq modèles afin de développer un *SIW* : *i)* un *modèle utilisateur* décrivant les besoins et profils des utilisateurs (individuel ou de groupe) ; *ii)* un *modèle de données* décrivant le domaine d'application supporté par le *SIW* ; *iii*) un *modèle d'accès progressif* décrivant les modalités d'accès progressif ; *iv)* un *modèle de fonctionnalités* décrivant les fonctionnalités du *SIW* (consultation, modification, *etc*.) et les aspects liés à la sécurité ; *v)* un *modèle hypermédia* décrivant les caractéristiques de présentation en termes de composition de pages Web et des aspects graphiques.

La notion d'*accès progressif* est un processus reposant sur le fait que l'utilisateur d'un *SIW* n'a pas besoin d'accéder à toute l'information tout le temps. L'*accès progressif* est alors utilisé pour construire un *SIW* qui a la capacité de fournir l'accès à ses ressources (c'est-à-dire, information et fonctionnalités) graduellement et d'une manière adaptée. Premièrement, les ressources considérées comme essentielles pour un utilisateur sont fournies, et ensuite, quelques-unes complémentaires, si elles sont nécessaires, sont proposées à travers une navigation guidée [Vill02]. En considérant un utilisateur nomade, il obtiendra tout d'abord seulement les ressources pertinentes (en tenant compte de sa localisation, ses besoins d'information, *etc*.).

Afin d'adapter l'information, il est nécessaire de définir le profil de l'utilisateur. Nous présentons des travaux qui le créent et le modélisent intégrant les caractéristiques de l'utilisateur et celles de son dispositif d'accès.

4.2. Représentation du profil de l'utilisateur

Selon Bouzeghoub *et al.* [Bouz05], le profil de l'utilisateur peut être vu comme un modèle personnalisé d'accès à l'information qui régit la manière de présenter les résultats du système. En ce qui concerne la définition du profil de l'utilisateur, plusieurs travaux ont été développés en considérant les centres d'intérêts de l'utilisateur [Zemi05] [Bouc06] [Kass05] [Tami06], son historique dans le système [Bouc06] [Tami06], ses besoins d'information [Zemi05], ses préférences [Bouz05] [Kech06] [Tami06] [Zemi05] afin de personnaliser l'information aux utilisateurs.

Tamine *et al.* [Tami06] et Zemirli *et al.* [Zemi05] présentent trois approches différentes de représentation du profil de l'utilisateur :

- *Approche ensembliste* : le profil est généralement formalisé sous forme de vecteurs de termes pondérés ou de classes de vecteurs. Le contenu est constitué d'un ou de plusieurs vecteurs définis dans un espace de termes. Ces termes sont obtenus à partir de plusieurs sources d'information concernant l'utilisateur. Les coordonnées des vecteurs correspondent aux poids associés aux termes retenus dans le profil. L'utilisation de plusieurs vecteurs correspond à deux préoccupations : prendre en compte des centres d'intérêt multiples et gérer leur évolution dans le temps ;
- *Approche sémantique* : la représentation du profil met en évidence, dans ce cas, les relations entre les contenus de l'information. Cette représentation se base sur l'utilisation d'ontologies ou de réseaux sémantiques probabilistes. Chaque catégorie de la hiérarchie représente la connaissance d'un domaine d'intérêt de l'utilisateur ;
- *Approche multidimensionnelle* : le profil est conçu comme un ensemble de dimensions, représenté selon divers formalismes, cherchant à modéliser/représenter l'utilisateur. Parmi les dimensions, on trouve les données personnelles, les centres d'intérêt, la qualité attendue des résultats délivrés, les préférences, *etc.*

Plusieurs travaux adoptent une approche multidimensionnelle pour représenter le profil de l'utilisateur. Bouzeghoub *et al.* [Bouz05] proposent ainsi un modèle générique de profils composé de six dimensions : les données personnelles, le centre d'intérêt, la qualité attendue, les préférences de livraison, la sécurité et l'historique des interactions de l'utilisateur. Tamine *et al.* [Tami06] et Zemirli *et al.* [Zemi05] s'appuient sur ces travaux et l'étendent en partie avec des données relatives à l'environnement logiciel, matériel et géographique dans lequel s'effectue la recherche de l'utilisateur. Kostadinov [Kost03] considère des aspects tels que les données personnelles, les centres d'intérêt, l'ontologie de domaine, la qualité attendue des résultats délivrés, la customisation, la sécurité et la confidentialité, le retour de préférences et les informations diverses. Zemirli *et al.* [Zemi05], en étudiant le travail de Kostadinov [Kost03], ont proposé un profil à trois dimensions (cf. Figure 4.1) : *i)* les données de préférences concernant les préférences de recherche et les centres d'intérêts de l'utilisateur ; *ii)* les données personnelles (son identité et sa profession) ; *iii)* les données sur l'environnement de travail de l'utilisateur (l'emplacement géographique, la configuration logicielle et matérielle). Les préférences de recherche représentent les préférences de l'utilisateur portant sur le processus de

recherche (temps de réponse), sur les documents recherchés (contenu et contenant) et sur la personnalisation des résultats (livraison et mise en page).

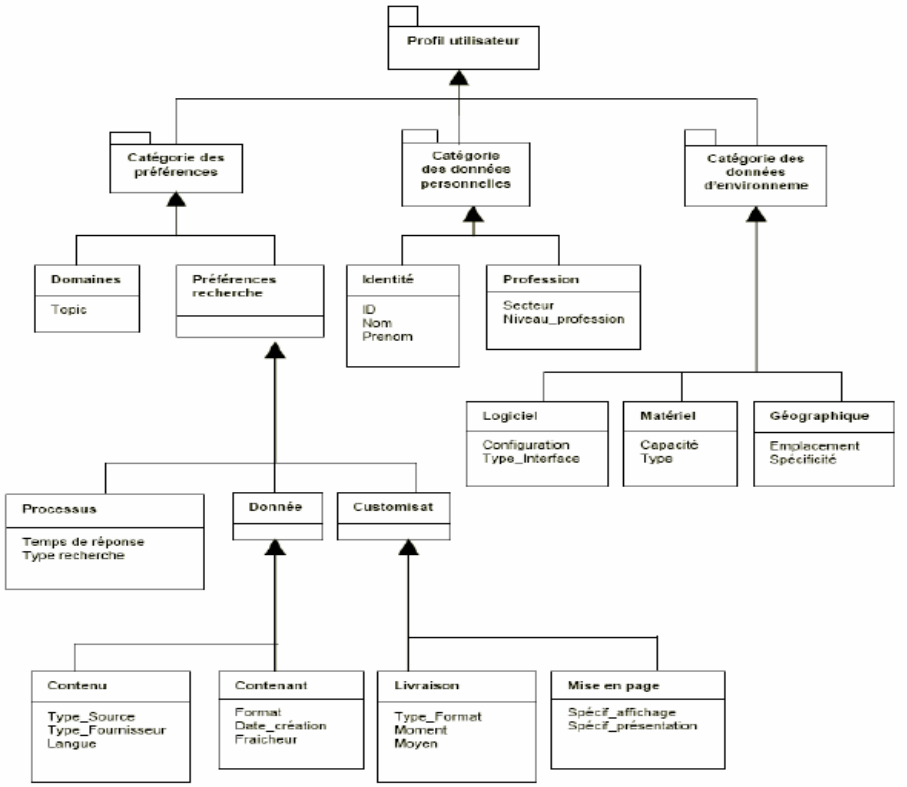


Figure 4.1. Représentation multidimensionnelle du profil utilisateur (*d'après Zemirli et al.* [*Zemi05*]).

Les préférences de l'utilisateur constituent un aspect très important pour la définition du profil de l'utilisateur : elles décrivent notamment les attentes de l'utilisateur par rapport au contenu et à la présentation de l'information dans son dispositif d'accès lors de son interaction avec le système. Dans la section suivante, nous présentons différents travaux sur la représentation des préférences de l'utilisateur.

4.3. Représentation des préférences de l'utilisateur

D'après Hafenrichter *et al.* [Hafe05] et Freuder *et al.* [Freu03], une préférence personnelle exprime les souhaits de l'utilisateur pendant ses interactions avec le système. Freuder *et al.* [Freu03] modélisent les préférences comme des contraintes légères où chaque combinaison de valeurs pour les variables est fournie par l'utilisateur et indique l'importance de cette contrainte dans le système. Hafenrichter *et al.* [Hafe05] modélisent les préférences comme des ordres partiels stricts de la manière suivante (en se basant sur le modèle de préférences de Kießling [Kieß05]) :

Soit $A = \{A_1, A_2, \ldots, A_k\}$, l'ensemble d'attributs A_j avec des domaines dom (A_j) où :

$dom(A) = dom(A_1) \times dom(A_2) \times ... \times dom(A_k)$

Une préférence *P* est un ordre partiel strict $P = (A, <_P)$, où $<_P \subseteq dom(A) \times dom(A)$

« $x <_P y$ » est interprété comme « je préfère *y* à *x* »

La personnalisation de requêtes demande un modèle de préférences de l'utilisateur sémantiquement riche, facile à gérer et flexible [Kieß05]. Koutrika *et al.* [Kout04] proposent un processus de personnalisation de requêtes, augmentant dynamiquement une requête avec des préférences de l'utilisateur stockées dans un profil afin de fournir des réponses personnalisées. Ce processus se base sur le modèle de préférences de Hafenrichter *et al.* [Hafe05], et sur des algorithmes pour la génération de réponses personnalisées en utilisant les préférences stockées. Le *modèle de préférences* repose sur des scores et un classement numérique des préférences provenant de profils structurés de l'utilisateur. Ce modèle représente et stocke des préférences dans des profils d'utilisateurs. Les *algorithmes* sélectionnent les préférences liées à une requête et génèrent progressivement des résultats personnalisés, classés selon les intérêts de l'utilisateur. Les préférences sont exprimées par les valeurs de leurs attributs et par les relations entre les entités indiquant dans quelle mesure les entités liées dépendent l'une de l'autre. Par exemple, *positif* signifie que l'utilisateur « *veut* », *négatif* « *il ne veut pas* » et *indifférent* « *cela lui est égal* ».

Belotti *et al.* [Belo04] considèrent que même, s'il n'y a pas de garantie que les préférences soient toujours satisfaites, elles doivent toujours appartenir au *contexte d'utilisation* puisqu'elles contribuent à la caractérisation dynamique de l'interaction de l'utilisateur avec le système. La section suivante est consacrée à la définition et à la caractérisation du *contexte d'utilisation*.

4.4. Représentation du contexte d'utilisation

D'après Jung *et al.* [Jung05], un contexte est considéré comme une source d'influences affectant la performance des systèmes d'information. Selon Indulska *et al.* [Indu03b], les systèmes pervasifs doivent être *sensibles au contexte* (c'est-à-dire, sensible à l'état de l'environnement et aussi aux besoins de l'application) car ces systèmes doivent pouvoir adapter l'information à l'utilisateur en considérant les capacités et les ressources de l'environnement courant. Zacarias *et al.* [Zaca05] et Belotti *et al.* [Belo04] considèrent que le *contexte* n'est pas une entité autonome : il est associé à l'information extraite de l'environnement physique (par exemple, à l'information spatiale telle que la localisation, l'orientation et la vitesse de déplacement) ainsi qu'à l'information environnementale (telle que la température, le niveau de lumière et de son, *etc*.). Selon Zacarias *et al.* [Zaca05], le contexte est vu comme une collection de ressources (propositions, suppositions, propriétés, procédures, règles, faits, concepts, *etc*.) associées à une situation spécifique (environnement, domaines, tâches, agents, interactions, conversations, *etc*.).

Pittarello [Pitt05] recense quelques-unes des caractéristiques contextuelles à considérer dans un processus d'adaptation telles que la localisation, le profil utilisateur, l'histoire de

l'utilisateur dans le système, le temps, le dispositif et le réseau. Schwinger *et al.* [Schw05], quant à eux, les regroupent en quatre types : *i)* celles liées à la portée (localisation, temps, dispositif, réseau et utilisateur) et à leur capacité de l'étendre ; *ii)* celles liées à la réutilisation du contexte en se basant sur des mécanismes d'inférence ou sur des profils prédéfinis ; *iii)* celles liées à l'acquisition du contexte (définissent le responsable de son acquisition – l'utilisateur, le système ou les deux – et le moment de l'acquisition - si le système acquiert le contexte lorsqu'il démarre ou pendant son exécution) ; *iv)* celles liées au mécanisme d'acquisition et d'utilisation du contexte (les mécanismes basés sur des techniques de « *push* » qui s'exécutent lorsqu'il y a des changements de contexte et ceux basés sur des techniques de « *pull* » qui s'exécutent lorsqu'une requête est formulée au système).

Dans leurs premiers travaux, Dey *et al.* [DeyA99] considéraient le contexte comme l'ensemble d'états physiques (localisation, la date, le temps), sociaux (les objets et les personnes dans l'environnement de l'utilisateur) et émotionnels (centres d'intérêt) de l'utilisateur. Dans des travaux plus récents, Dey *et al.* considèrent que :

« *Le contexte est construit à partir de tous les éléments d'information qui peuvent être utilisés pour caractériser la situation d'une entité. Une entité correspond ici à toute personne, tout endroit, ou tout objet (en incluant les utilisateurs et les applications eux-mêmes) considéré(e) comme pertinent(e) pour l'interaction entre l'utilisateur et l'application.* »[59]

Selon Dey *et al.* [DeyA99] :

« *Un système est sensible au contexte s'il utilise du contexte pour fournir de l'information pertinente et/ou des services à l'utilisateur, où la pertinence dépend sur la tâche de l'utilisateur* »[60].

Ces auteurs classent les applications sensibles au contexte en trois catégories : *i)* celles qui présentent de l'information et des services à l'utilisateur ; *ii)* celles qui exécutent automatiquement des services (des services exécutés automatiquement compte tenu du contexte basé sur des règles) ; *iii)* celles qui possèdent la capacité d'associer des données avec le contexte de l'utilisateur.

En s'appuyant sur la définition de *contexte* de Dey *et al.* [DeyA99], Kirsch-Pinheiro [Kirs06] propose une représentation par objets du *contexte d'utilisation* d'un utilisateur nomade dans un collecticiel sur le Web. Afin d'identifier les éléments pertinents du contexte d'utilisation, Kirsch-Pinheiro [Kirs06] a pris en compte cinq points de vue différents : *espace* (où ?), *outils* (comment ?), *temps* (quand ?), *communauté* (qui ?) et *processus* (quoi ?) qui concernent l'endroit, la manière, le moment, les participants et le processus de coopération liés à l'interaction entre l'utilisateur et le système dans le cadre du travail coopératif. Chaque point de

[59] La définition originale en anglais est : « *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves* ».

[60] La définition originale en anglais est : « *A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task* ».

vue fait référence à certains éléments du *contexte d'utilisation* : l'élément localisation réfère au point de vue *espace*, l'application et le dispositif à l'*outil*, un calendrier partagé au *temps*, les éléments *groupe*, *rôle* et *membre* au point de vue de *communauté* et finalement, le processus, les activités et les objets partagés au point de vue de *processus*. La Figure 4.2 montre les diagrammes de classes qui représentent les éléments constituant le *contexte d'utilisation* selon ce travail :

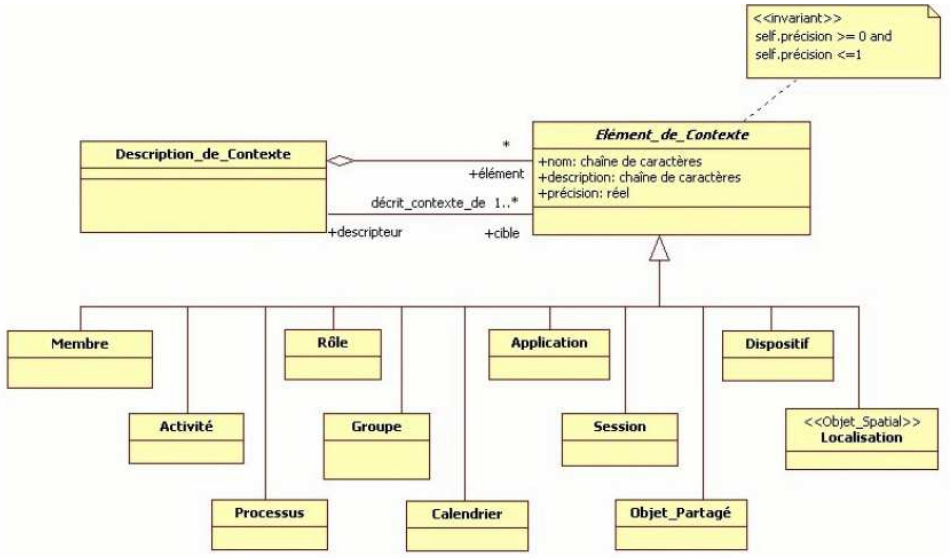


Figure 4.2. Diagramme de classes de la représentation de contexte d'utilisation d'un utilisateur nomade dans un collecticiel (*D'après Kirsch-Pinheiro* [*Kirs06*]).

D'après Bucur *et al.* [Bucu06], les deux principaux problèmes des applications sensibles au contexte concernent d'un côté, la quantité d'information contextuelle qui doit être gérée, et de l'autre, la pertinence de cette information (comment définir la pertinence, comment choisir entre information pertinente et non pertinente et comment utiliser cette information pertinente). Leur travail décrit le contexte comme un groupe d'informations pertinentes pour une finalité spécifique. Une finalité (but à atteindre) est l'information la plus intéressante pour l'application à un moment donné (par exemple, décider l'action à suivre lors d'une proposition, expliquer une action, comprendre une conversation, *etc.*). Les finalités déterminent la manière dont l'application prendra en considération le contexte et agira en l'utilisant. Le contexte est composé d'attributs représentant des éléments d'information. Chaque attribut a toujours une valeur à un moment donné, valeur qui dépend de plusieurs entités liées par cet attribut. Une entité est une instance de personne, objet, lieu, activité ou concept organisationnel (par exemple, rôle, groupe, *etc*.). Par exemple, on peut définir une entité « Personne » qui possède comme propriétés « intérêts », « activitéCourante », *etc*. Les entités peuvent être définies en utilisant une

hiérarchie de classe[61] (cf. Figure 4.3). Par exemple, « Personne » et « Groupe » sont des « Entités » « Sociales ». Un attribut de contexte « PersonneEstMembreDe » prend pour entrée une « Personne » et retourne les « Groupes » auxquels cette « Personne » appartient.

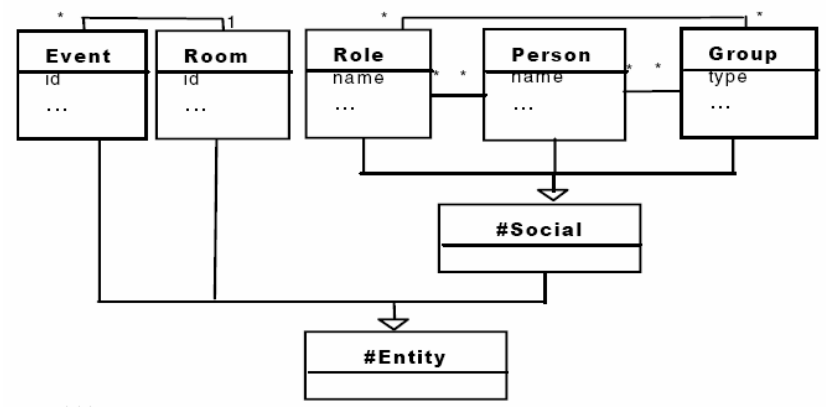


Figure 4.3. Hiérarchie des entités et leurs propriétés (*D'après Bucur* [Bucu06]).

Bucur *et al.* [Bucu06] mentionnent plusieurs manières de représenter le contexte telles que les graphes contextuels, XML, ou les modèles orientés par objets, entre autres. D'après Bucur *et al*, toutes ces représentations souffrent d'un manque de généralité : la plupart sont restreintes à un type spécifique d'application et expriment une vue particulière d'un contexte. Ces représentations manquent aussi de bases formelles nécessaires pour capturer le contexte d'une manière consistante et pour supporter des raisonnements sur leurs différentes propriétés. Pour ces raisons, Bucur *et al.* utilisent deux ontologies afin de définir les éléments qui composent le contexte et sa finalité : une ontologie de contexte (pour définir tous les attributs de contexte) et une ontologie de domaine (pour définir tous les concepts qui seront utilisés par le système).

Dans la section suivante, nous présentons des travaux concernant l'adaptation réalisée à l'aide d'agents, compte tenu des caractéristiques de l'utilisateur, du contexte d'utilisation et des contraintes techniques de son dispositif d'accès.

4.5. Adaptation à l'aide d'agents

Plusieurs aspects techniques et fonctionnels doivent être considérés lors de la conception d'un *SIW* accédé à travers des *DM*, notamment l'adaptation de l'information délivrée à l'utilisateur nomade [Niet04] [Rahw04] [Talm05] [Wool95]. Nous présentons ici quelques architectures ou *frameworks* basés sur des agents qui adaptent l'information aux utilisateurs compte tenu du profil de l'utilisateur, de la confidentialité de l'information, de l'accès aux ressources et services dans un environnement ubiquitaire et des caractéristiques d'un groupe de travail.

[61] Toutes les classes héritent de la classe *Entité*.

4.5.1 Architectures qui adaptent l'information

Compte tenu du profil de l'utilisateur, Kurumatani [Kuru04] et Berhe *et al.* [Berh04] proposent des architectures afin d'adapter le contenu de l'information à un utilisateur. Kurumatani [Kuru04] propose *CONSORTS*, une architecture basée sur des agents ubiquitaires et conçue pour un support massif de *DM* décrit précédemment (cf. section 3.2). Berhe *et al.* [Berh04] proposent une architecture qui exploite quatre profils afin d'adapter l'information : *i)* le *profil de contenu ou média* (qui gère l'information correspondant aux type, format, taille, et localisation du media) ; *ii)* le *profil de l'utilisateur* (qui gère l'information sur les préférences) ; *iii)* le *profil du dispositif* (qui gère l'information sur les capacités matérielles et logicielles) ; *iv)* le *profil de réseau* et *services* (qui gère l'information sur les formats de médias supportés, la connexion de réseau, la bande passante, la latence, les performances). Aucune de ces architectures ne considère la récupération de l'information stockée dans différents types de dispositifs (serveurs et *DM*), ni la manière dont le profil de l'utilisateur est défini et spécifié. Elles ne précisent pas la manière de représenter et de gérer les *préférences de l'utilisateur* ni le *contexte d'utilisation* de la session courante.

Concernant la confidentialité de l'information, Gandon *et al.* [Gand04] et Titkov *et al.* [Titk04] proposent respectivement une *architecture* basée sur le Web Sémantique, sensible au contexte et qui tient compte des préférences de confidentialité (« *privacy preferences* ») de l'utilisateur, et un *framework* prenant en compte la confidentialité des données de l'utilisateur dans les applications ubiquitaires. L'architecture de Gandon *et al.* [Gand04] supporte l'accès et la découverte automatique de ressources personnelles de l'utilisateur (telles que ses fichiers, son agenda, *etc*.) accessible en fonction des préférences de confidentialité spécifiées par l'utilisateur. Les règles d'invocation des services (à l'aide des ontologies de services et de profils de services) permettent d'identifier les ressources disponibles les plus pertinentes en réponse à une requête. Cependant, ce travail ne tient pas compte du fait que l'information constituant la réponse à une requête peut être distribuée entre différentes sources. Le *framework* de Titkov *et al.* [Titk04] considère des aspects tels que l'*ubiquité* (être disponible n'importe où), l'*universalité* (être capable de fonctionner dans des environnements hétérogènes), l'*unicité* (liée aux caractéristiques d'un contexte particulier telles que la localisation) et le travail à l'*unisson* (permettant aux multiples parties de travailler ensemble). Néanmoins, Titkov *et al.* ne précisent pas la manière d'accomplir chacun de ces aspects.

Albayrak *et al.* [Alba05], Calisti *et al.* [Cali04] et Sashima *et al.* [Sash04] proposent également des solutions basées sur des agents qui tiennent compte des caractéristiques d'accès aux ressources et services dans des environnements ubiquitaires. Dans la suite, nous décrivons chacun de ces travaux :

PIA [Alba05] est un *Système d'Information* basé sur des agents personnels pour collecter, filtrer et intégrer l'information en offrant l'accès à l'information aux clients *WWW*, e-mail, *SMS*, *MMS* et *J2ME*. *PIA* combine des techniques de « *push* » et « *pull* » afin de permettre à l'utilisateur d'une part, de chercher explicitement l'information spécifique, et d'autre part, d'être automatiquement informé sur l'information pertinente par rapport à des activités prévues pour la journée. Cependant, le *système PIA* cherche seulement l'information en format texte. Il ne prend

donc pas en compte l'adaptation de différents types de médias selon différents *DM*. De plus, ce système ne gère pas l'adaptation à la localisation de l'utilisateur.

Calisti *et al.* [Cali04] présentent une proposition offrant aux utilisateurs nomades un accès dynamique et adaptatif à une variété de services de communication, à travers l'amélioration de la connectivité au réseau et l'optimisation de l'utilisation des ressources. Cette proposition exécute les agents sur différents dispositifs et éléments de réseau (en gérant l'espace ubiquitaire), tient compte des changements de l'environnement ubiquitaire (sensible au contexte), en adaptant le système aux comportements de l'utilisateur, réagit et anticipe les changements des besoins de l'utilisateur (interfaces flexibles de services). Egalement, cette proposition fournit des services unitaires et globaux conçus en considérant les préférences et contraintes des différents utilisateurs (coordination et négociation dynamiques). Néanmoins, ce travail ne prend pas en compte l'adaptation de différents types de médias selon différents *DM*, ni ne précise les mécanismes de représentation du *contexte d'utilisation*.

Sashima *et al.* [Sash04] proposent un *framework* pour la coordination des services et des dispositifs afin de satisfaire les besoins d'information d'un utilisateur nomade. Ce *framework* assiste les utilisateurs qui accèdent à des ressources dans des environnements ubiquitaires. Ces auteurs considèrent les caractéristiques contextuelles d'un utilisateur nomade, en particulier, sa localisation. Néanmoins, ce *framework* ne considère pas l'adaptation de l'information selon différents dispositifs d'accès ni la distribution des données entre différents dispositifs.

Afin de permettre l'interaction, la création et la gestion de groupes (qu'ils appellent « *sociétés* »), Kamara *et al.* [Kama04] proposent une architecture basée sur des agents qui possèdent des caractéristiques cognitives (liées à la représentation de la connaissance et le raisonnement) et communicatives (liées aux protocoles de communication). Les activités des agents à l'intérieur de telles sociétés sont réglementées en utilisant des langages dédiés. Leur approche permet de gérer la connaissance des agents (capacités cognitives), d'organiser des communautés d'agents (organisation sociale et régulatrice), de coordonner et de contrôler leurs tâches (individuelles et/ou collectives). De plus, l'architecture propose pour un agent, un protocole d'*admission* pour créer, rejoindre ou quitter une société. Un protocole de *gestion de session* est également fourni pour grouper les agents qui travaillent ensemble en tâches communes, et des protocoles orientés tâches pour coordonner et contrôler les activités d'un agent jouant différents rôles en différents groupes. Néanmoins, cette architecture ne considère pas la distribution de l'information entre *DM*, ni ne précise les mécanismes d'adaptation à utiliser. Rien n'est mentionné sur l'existence d'une adaptation de l'information au niveau individuel ou collectif.

Lech *et al.* [Lech05] proposent *AmbieAgents*, une infrastructure basée sur des agents pour la livraison d'information aux utilisateurs nomades sensible au contexte. Pour *AmbieAgents*, le contexte est une représentation à l'intérieur de l'ordinateur des aspects d'une situation dans le monde réel. Ce contexte est composé de cinq sous-contextes : le *social*, *de tâches*, *personnel*, *de l'environnement* et *des caractéristiques spatio-temporelles*. Chacun de ces contextes est représenté comme un ensemble structuré de paires « *attribut-valeur* » capturant des aspects pertinents d'une situation de l'utilisateur, où la pertinence dépend du domaine d'application. Par

exemple, la localisation est un aspect plus important pour une application de transport que pour une application de publicité. *AmbieAgents* a été implémenté pour une application qui définit le profil des voyageurs à l'aéroport international d'Oslo, en fonction de leurs préférences pour les repas et les achats (contexte personnel), la localisation de l'utilisateur (contexte des caractéristiques spatio-temporelles), l'état du vol de l'utilisateur (contexte de l'environnement) et l'activité que l'utilisateur veut accomplir (contexte de tâches, pour cette application l'activité – *manger* ou *acheter* - peut être fournie par l'utilisateur ou inférée par le système en prenant en compte la localisation courante de l'utilisateur). Harvey *et al.* [Harv05] ont proposé à travers *MADSUM,* un système adaptatif distribué de support à la prise de décision qui utilise d'une part, un processus de négociation afin de solliciter et d'organiser les agents pour produire de l'information, et d'autre part, un processus assemblant d'une manière cohérente l'information pour la prise de décision. Ce système définit un modèle de l'utilisateur contenant de l'information sur l'utilisateur, ses préférences, ses contraintes et ses priorités. Le système de support à la prise de décision se base sur quatre activités :

- Le système envoie une fonction d'utilité et l'information concernant l'utilisateur à tous les agents ;
- Les agents génèrent des estimations exprimées comme un ensemble d'attributs ;
- Chaque agent renvoie ses estimations à ses enfants dans la hiérarchie ;
- Chaque agent compile, intègre et propage les résultats obtenus vers ses ancêtres dans la hiérarchie.

4.5.2 Techniques pour l'adaptation dans des Systèmes Multi-Agents

Afin d'adapter l'information aux centres d'intérêt et aux préférences des utilisateurs, des techniques pour des systèmes et des applications basées sur des agents ont été proposées :

L'approche de Cole *et al.* [Cole05] applique des techniques d'*apprentissage automatique* (« *Machine Learning* ») pour créer des agents qui facilitent l'interaction entre un utilisateur et l'Internet, et pour personnaliser le comportement des agents en prenant en compte les intérêts et les préférences de l'utilisateur. Cette approche à été évaluée dans un système de recommandation de programmes de télévision, de films, de musique, et pour chercher l'information sur Internet. Les techniques utilisées permettent à l'agent de découvrir les intérêts et les préférences de l'utilisateur, et d'expliquer à l'utilisateur la raison pour laquelle il fait une recommandation. D'autre part, elles permettent à l'utilisateur d'utiliser un langage d'hypothèse afin d'établir des règles sur le sujet pour lequel il souhaite disposer de recommandations.

Talman *et al.* [Talm05] proposent un modèle caractérisant l'aide fournie par un agent en termes de « *coopération* » et de « *fiabilité* ». Un agent choisit une action en estimant le degré d'aide apporté par d'autres agents, en fonction de relations de dépendance entre agents. Ce modèle a été évalué dans un jeu de négociation dans lequel les joueurs échangent des ressources pour atteindre leurs buts. Les agents doivent être capables d'identifier et de négocier avec ceux qui sont coopératifs et d'éviter ceux qui sont exploiteurs. Le modèle de Talman *et al*. utilise la théorie des jeux afin de guider le comportement des agents. Cette théorie fournit aux agents des

stratégies pour tenir compte des effets de leurs décisions sur chacun des autres agents et elle garantit qu'aucun agent n'est exploité par un autre. Tous les agents délibèrent de la même manière sur le jeu et suivent les stratégies prescrites. Le modèle caractérise la personnalité d'un agent à travers deux dimensions : la *coopération* (tendance à proposer des échanges de ressources qui apportent mutuellement du bénéfice) et la *fiabilité* (tendance à garantir des résultats).

L'estimation de l'état d'un agent consiste en la mise à jour des croyances d'un agent étant donné un ensemble d'actions exécutées et de l'évidence observée (généralement obtenue à travers des capteurs). Doshi *et al.* [Dosh05] modélisent la croyance d'un agent en utilisant une distribution probabiliste et en estimant l'état de l'agent en utilisant des *filtres bayésiens.* Un *filtre bayésien* permet à l'agent de maintenir une croyance sur l'état du monde à un moment donné et de le mettre à jour chaque fois qu'une action est accomplie et qu'une nouvelle information (qui peut le modifier) arrive. La croyance d'un agent récapitule toute l'information contenue en actions et observations passées. L'estimation de l'état d'un *SMA* consiste à stocker l'histoire des croyances de ses agents et de leurs actions qui peuvent produire des changements de l'état ou des croyances.

Maximilien *et al.* [Maxi05] proposent un *framework* basé sur des agents dans lequel les agents considèrent les préférences de *Qualité de Service* (« *QoS* ») des consommateurs, déterminent les niveaux de confiance associés aux fournisseurs, et sélectionnent les services en représentation des consommateurs. Les agents de services sont en charge de sélectionner le(s) « meilleur(s) » service(s) pour le consommateur en utilisant un modèle de confiance (« *trust* ») qui repose sur une conceptualisation partagée (ontologie) de *QoS* et un modèle de préférences de *QoS* des consommateurs. La confiance d'un service prédit sa qualité pour une utilisation future, et peut être estimée à partir de l'histoire des niveaux de sa qualité (cette histoire correspond à sa *réputation*).

Birukov *et al.* [Biru05] proposent « *Implicit* », un système de *Recommandation* pour supporter des communautés de personnes possédant les mêmes intérêts en cherchant l'information sur le web à travers des moteurs de recherche comme *Google*. *Implicit* se base sur les concepts d'*Implicit Culture*, une généralisation des *filtres collaboratifs*[62], qui s'appuie sur l'hypothèse qu'un nouveau membre d'une communauté se comporte similairement aux autres membres sans qu'il soit nécessaire d'exprimer explicitement la connaissance de la communauté. Lorsqu'un utilisateur formule une requête, *Implicit* suggère l'information spécifique en exploitant des observations précédentes sur le comportement d'autres utilisateurs qui ont formulé des requêtes similaires. Chaque utilisateur possède son agent personnel capable d'interagir avec les agents personnels d'autres utilisateurs. Le système implémente une approche collaborative qui fournit à l'utilisateur des suggestions d'autres utilisateurs et les résultats fournis par les moteurs de recherche. Les agents utilisent des techniques de « *Data Mining* » afin d'apprendre et de découvrir les comportements des utilisateurs et les agents

[62] Les *filtres collaboratifs* sont une technique de production de recommandations personnelles qui utilisent des similarités entre des classements d'utilisateurs.

interagissent pour partager la connaissance sur leurs utilisateurs. Ce système de recommandation accepte des requêtes d'un utilisateur et exploite la connaissance sur ses besoins, ses comportements, ses profils de recherche et le contenu de l'information afin de fournir des recommandations personnalisées sur des sujets particuliers.

4.6. Conclusion

Ce chapitre a été consacré à l'adaptation, un aspect fondamental pour fournir à l'utilisateur une information utile qui tient compte de ses besoins, préférences, et des caractéristiques de son dispositif d'accès. Toutes ces caractéristiques sont contenues dans le *contexte d'utilisation.*

Tout d'abord, nous avons présenté *l'adaptation au niveau du contenu* et *l'adaptation au niveau des liens* en soulignant leurs principales caractéristiques. Puis, nous avons introduit les *systèmes hypermédias adaptatifs* qui tiennent compte de ces deux types d'adaptation mais qui ne considèrent pas le besoin d'adapter l'information aux caractéristiques des environnements nomades telles que la *localisation* (qui peut modifier les besoins d'information de l'utilisateur), et les caractéristiques du *dispositif d'accès mobile* (qui peut contraindre l'affichage de l'information). Nous avons aussi décrit les principaux modèles sur lesquels repose l'adaptation de l'information.

Ensuite, nous nous sommes intéressés aux différents modèles existants pour la représentation du profil de l'utilisateur, de ses préférences et du contexte d'utilisation. Pour le *profil de l'utilisateur*, nous avons présenté trois approches différentes : une approche *ensembliste* (basée sur des vecteurs), une approche *sémantique* (qui explicite les relations entre les contenus de l'information) et une approche *multidimensionnelle* (qui considère un ensemble de dimensions). Aucune de ces propositions ne présente de façon détaillée les mécanismes permettant d'exploiter la représentation du *contexte d'utilisation*, ni celle des *préférences d'un utilisateur nomade.*

En ce qui concerne la représentation des *préférences de l'utilisateur*, les travaux de Hafenrichter *et al.* [Hafe05], Freuder *et al.* [Freu03], Kießling [Kieß05], Koutrika *et al.* [Kout04] ou encore de Belotti *et al.* [Belo04] sont des propositions originales. Néanmoins, ces travaux ne précisent pas la manière de représenter les souhaits de l'utilisateur à propos : *i)* des activités qu'il souhaite mener, *ii)* des contenus que le système lui délivre et, *iii)* de l'affichage des informations.

En termes de *représentation du contexte d'utilisation*, nous avons présenté des travaux de [Jung05] [Indu03b] [Zaca05] [Pitt05] [Schw05] basés sur le travail de Dey *et al.* [DeyA99], et qui intègrent des caractéristiques des environnements nomades. Plusieurs travaux proposent des représentations du *contexte d'utilisation* d'un utilisateur nomade. Parmi ces travaux, celui de Kirsch-Pinheiro[Kirs06] concerne une représentation orientée objets prenant en compte l'endroit, la manière, le moment, les participants et le processus de coopération référant à l'interaction entre l'utilisateur et le système dans le cadre d'un travail coopératif, tandis que la représentation de Bucur *et al.* [Bucu06] utilise des ontologies pour définir tous les attributs du contexte et tous les concepts qui seront utilisés par le système.

D'autres travaux basés sur des agents tels que ceux présentés par Albayrak *et al.* [Alba05], Kurumatani [Kuru04] et Sashima *et al.* [Sash04] constituent des propositions intéressantes de modélisation du contexte en vue d'adapter l'information à un utilisateur nomade. Cependant, ces propositions ne sont pas entièrement satisfaisantes. Par exemple, dans Kurumatani [Kuru04] certaines caractéristiques de l'utilisateur (telles que ses droits d'accès, sa localisation, les caractéristiques de son *DM*, ses préférences) ne sont pas prises en compte. Le système *PIA* [Alba05] limite son traitement de l'information au format texte et ne considère pas la localisation de l'utilisateur. D'autres auteurs, comme Albayrak *et al.* [Alba05] et Sashima *et al.* [Sash04] ne considèrent pas l'adaptation de l'information aux *DM*. Des travaux tels que *MADSUM* [Harv05] et *AmbieAgents* [Lech *05*] présentent des mécanismes explicites pour personnaliser l'information à l'utilisateur en considérant les *préférences de l'utilisateur* dans le cas de *MADSUM*, et le *contexte d'utilisation*, dans le cas d'*AmbieAgents*.

Finalement, des techniques pour des systèmes et des applications basées sur des agents, telles que l'*apprentissage automatique* [Cole*05*], la *négociation et* la *coopération* [Talm05], l'*apprentissage bayésien* [Dosh05], la *qualité de service* [Maxi05] et la *recommandation* [Biru05] ont été proposées afin d'adapter l'information par rapport aux centres d'intérêts et aux *préférences de l'utilisateur*. Cependant, ces propositions ne présentent pas clairement la représentation du *contexte d'utilisation* ni celle des *préférences de l'utilisateur* sur lesquelles elles reposent.

Le chapitre suivant présente une synthèse de l'état de l'art qui analyse les contributions de différents travaux des trois domaines étudiés (l'*informatique ubiquitaire*, les *Systèmes Multi-Agents* de l'*adaptation*), par rapport à notre problématique d'intérêt qui concerne l'accès aux *SIW* à travers des *DM* et l'adaptation de l'information dans des environnements nomades.

5. SYNTHESE

Ce chapitre propose une synthèse dont le but est d'évaluer dans quelle mesure des travaux présentés dans l'état de l'art contribuent à l'accès et à l'adaptation de l'information dans des environnements ubiquitaires. Cette synthèse est organisée de la manière suivante : la section 5.1 présente une analyse portant sur l'utilisation d'agents pour l'accès aux *SIW* à travers des *DM*. Cette analyse montre les besoins d'adaptation de l'information à prendre en compte (adaptation par rapport aux caractéristiques de l'utilisateur et à celles de son dispositif d'accès). La section 5.2 présente des travaux liés à la représentation des préférences de l'utilisateur et de son contexte d'utilisation que nous considérons comme les aspects de base pour la définition du profil de l'utilisateur. Des travaux basés sur des agents, qui adaptent l'information dans des environnements nomades, sont aussi traités. La section 5.3 présente la contribution de notre proposition qui fournit des solutions aux problèmes et contraintes détectés pendant l'analyse de ces travaux.

5.1. Agents pour l'accès aux Systèmes d'Information à travers des DM

Parmi les architectures basées sur des agents qui permettent aux utilisateurs d'accéder aux systèmes d'information à travers des *DM*, nous avons particulièrement étudié : *KODAMA*, *MIA* et *CONSORTS* dont les caractéristiques sont résumées dans le Tableau 2 :

	***KODAMA* [Taka03]**	***MIA* [Beus00]**	***CONSORTS* [Kuru04]**
Distribution des données	plusieurs serveurs	plusieurs serveurs	serveur de contenu
Protocoles de communication	*TCP/IP*	*http, WAP*	*http*
Types de données multimédia	texte	texte	texte, images
Aspects pris en compte pour la définition du profil	*non définis*	*préférence et localisation de l'utilisateur*	*préférences, intentions et caractéristiques*
Type de DM	*téléphone portable*	*PDA, téléphone portable, téléphone portable WAP*	*PDA, téléphone portable, ordinateur portable*
Mécanisme de détection de localisation	transmetteur dans le *DM* et récepteurs dans les lieux	GPS ou localisation entrée par l'utilisateur	capteur : camera ou LAN sans fil

Tableau 2. Architectures pour modéliser SMA.

Les trois premiers aspects du Tableau 2 concernent les caractéristiques de distribution de l'information (*KODAMA* et *MIA* reposent sur une architecture distribuée tandis que *CONSORTS* est basé sur une architecture centralisée) et les types de données multimédias gérées par les sources d'information (les trois traitent essentiellement du texte). Les trois derniers aspects sont ceux à considérer pour l'adaptation de l'information. Nous pouvons conclure que même si *MIA* et *CONSORTS* prennent en compte les préférences de l'utilisateur, ni les mécanismes de représentation et de gestion de telles préférences ni les mécanismes d'adaptation utilisés ne sont décrits. Toutes les trois gèrent l'accès à travers des *DM* mais aucune ne présente des mécanismes d'adaptation de l'information en intégrant des caractéristiques de tels dispositifs. Finalement, ces trois architectures possèdent des mécanismes de détection de la localisation mais elles ne précisent ni le rôle, ni la pertinence de la localisation pour l'adaptation de l'information.

Malgré les limitations de ces architectures par rapport aux mécanismes d'adaptation, nous avons identifié trois niveaux généraux, communs à de telles architectures, qui permettent aux utilisateurs d'accéder et de formuler des requêtes aux *SIW* à travers des *DM* :

- Un *niveau des agents mobiles* qui est composé du *DM* et des *agents mobiles*. L'utilisateur accède au système à travers son *DM* sur lequel les *agents mobiles* s'exécutent ;
- Un *niveau intermédiaire* qui offre des services (de connexion, de communication, *etc.*) afin de permettre l'interaction entre les deux autres niveaux ;
- Un *niveau de système d'information* qui représente les services (en tant que réponses à des besoins fonctionnels) que le système fournit aux utilisateurs.

Niveau	Correspond à	Il manque
agents mobiles	Caractéristiques de Connexion	L'adaptation de l'information en considérant la localisation de l'utilisateur et le moment de connexion du *DM*
intermédiaire	Caractéristiques de Communication	L'adaptation de l'information selon les caractéristiques du *DM*
système d'information	Caractéristiques de Gestion d'Information	L'adaptation de l'information selon les caractéristiques et préférences de l'utilisateur

Tableau 3. Défauts des trois niveaux communs pour les architectures basées sur des agents.

Chaque niveau contribue à l'accès aux *SIW* à travers des *DM* mais aucun ne considère des caractéristiques envisageant l'adaptation de l'information (cf. Tableau 3). Le *niveau d'agents mobiles* devrait considérer la manière d'obtenir des caractéristiques spatio-temporelles pour adapter l'information par rapport à des caractéristiques contextuelles des environnements nomades. Le *niveau intermédiaire* devrait considérer le type de *DM* connecté et ses caractéristiques pour adapter l'affichage de l'information sur le dispositif d'accès. Le *niveau de système d'information* devrait considérer les caractéristiques et préférences de l'utilisateur pour adapter le contenu de l'information.

La section suivante est consacrée à l'analyse de travaux qui adaptent l'information en considérant les caractéristiques de l'utilisateur et celles de son dispositif d'accès. Nous présentons aussi des travaux ciblés sur la représentation du *contexte d'utilisation*, des *préférences* et *profil de l'utilisateur*.

5.2. Adaptation de l'information

D'après Brusilovsky [Brus00] et Henze *et al*. [Henz02], l'information peut être adaptée par rapport à deux niveaux :

- Au *niveau du contenu* qui consiste à adapter le contenu d'une page hypermédia aux préférences, buts, connaissances et d'autres informations stockées dans le modèle de l'utilisateur ;
- Au *niveau de la navigation* qui aide les utilisateurs à trouver leurs chemins dans l'hyperespace en adaptant la présentation et les fonctionnalités des liens aux buts, connaissances et d'autres caractéristiques de l'utilisateur.

Les *systèmes hypermédias adaptatifs* sont un exemple de systèmes qui prennent en compte les deux niveaux précédents pour adapter l'information. Ces systèmes construisent un modèle de buts, de préférences et de la connaissance de chaque utilisateur, qui est exploité lors de l'interaction avec l'utilisateur afin d'adapter l'information à ses besoins [Brus00]. Cependant, ces systèmes ne spécifient pas la manière d'adapter l'information aux caractéristiques des environnements nomades (telles que la localisation et le moment de connexion), ni aux caractéristiques du *DM*.

Le profil de l'utilisateur peut être vu comme un modèle personnalisé d'accès à l'information qui régit la manière de présenter les résultats des systèmes. Plusieurs approches ont été proposées pour sa représentation : l'approche *ensembliste* qui formalise le profil sous forme de vecteurs de termes pondérés ou de classes de vecteurs ; l'approche *sémantique* qui permet au profil de mettre en évidence les relations entre les contenus de l'information ; l'approche *multidimensionnelle* qui conçoit le profil comme un ensemble de dimensions cherchant à représenter l'utilisateur. Nous nous intéressons à la représentation multidimensionnelle du profil. Le Tableau 4 recense des travaux qui représentent le profil de l'utilisateur en considérant plusieurs dimensions.

Les conventions utilisées pour ce tableau sont : (+) pour indiquer que cet aspect est pris en compte dans le travail, (?) pour indiquer que cet aspect n'est pas précisé dans le travail (c'est-à-dire, la manière dont il est traité n'est pas précisée) et (-) pour indiquer que cet aspect n'est pas explicitement pris en compte.

	Zemirli *et al.* [Zemi05]	Bouchard *et al.* [Bouc06]	Kassab *et al.* [Kass05]	Tamine *et al.* [Tami06]	Bouzeghoub *et al.* [Bouz05]	Kechid *et al.* [Kech06]
Données personnelles	+	+	+	+	+	+
Centres d'intérêts	+	+	+	+	+	?
Historique dans le système	+	+	-	+	+	-
Besoins d'information	+	-	-	?	+	?
Préférences de l'utilisateur	+	?	?	+	+	+

Tableau 4. Travaux qui représentent un profil de l'utilisateur selon plusieurs dimensions.

La plupart de ces travaux montrent un profil dont les dimensions sont les données personnelles, les centres d'intérêts et l'historique de l'utilisateur dans le système. Plusieurs travaux qui considèrent les préférences de l'utilisateur comme une dimension d'adaptation, les réduisent à des expressions sur la manière de présenter les données à l'utilisateur. Ces préférences ne correspondent pas aux activités que l'utilisateur veut accomplir dans le système ni aux résultats attendus de ces activités.

Nous avons étudié différentes approches pour la représentation des *préférences de l'utilisateur* (concernant les activités de l'utilisateur, leurs résultats attendus et la manière de les afficher sur le dispositif d'accès). Nous avons présenté des travaux tels que ceux de Hafenrichter *et al.* [Hafe05], Freuder *et al.* [Freu03], Kießling [Kieß05], Koutrika *et al.* [Kout04] et Belotti *et al.* [Belo04]. Néanmoins, ces travaux ne précisent pas la manière de représenter les souhaits de l'utilisateur à propos : *i)* des activités qu'il a à mener ; *ii)* des contenus que le système lui délivre et *iii)* de l'affichage de ces contenus.

Concernant l'adaptation de l'information pour des environnements nomades, nous avons présenté différents travaux basés sur des agents (cf. section 4.5.1). Le Tableau 5 résume sept

aspects relatifs à l'adaptation et leur prise en compte dans les travaux présentés. Nous considérons ainsi :

- *Le profil de l'utilisateur* : la contribution utilise et/ou présente des mécanismes pour définir le profil de l'utilisateur ;
- *La localisation de l'utilisateur* : la contribution tient compte de la localisation de l'utilisateur pour adapter l'information, pour définir le profil de l'utilisateur, pour traiter des requêtes, *etc.* ;
- *Les préférences de l'utilisateur* : la contribution présente des mécanismes de représentation et/ou gestion de préférences. Le travail prend en compte les préférences de l'utilisateur dans un but d'adaptation ;
- *L'adaptation aux différents types de média* : la contribution utilise et/ou présente des mécanismes pour adapter l'affichage de différents types de média ;
- *Les relations entre les agents* : la contribution présente l'organisation des agents qui interviennent, leurs rôles et leurs activités ;
- *Les mécanismes de routage de requêtes* : la contribution présente ou mentionne des algorithmes, des mécanismes ou des processus de routage de requêtes, afin de rediriger les requêtes vers les systèmes d'information les plus pertinents pour y répondre (en considérant ou non, des critères d'adaptation) ;
- *La distribution d'information entre DM* : la contribution présente et/ou mentionne des mécanismes pour obtenir l'information provenant de différentes sources d'information, notamment, si ces sources sont stockées dans un *DM*.

Les conventions utilisées pour le Tableau 5 sont : (+) pour indiquer que cet aspect est pris en compte dans le travail. (-) pour indiquer que cet aspect n'est pas explicitement pris en compte, et (?) pour indiquer que cet aspect n'est pas précisé dans le travail (c'est-à-dire, la manière dont il est traité n'est pas précisée).

	Kurumatani et al. [Kuru04]	Berhe et al. [Berh04]	Gandon et al. [Gand04]	Titkov et al. [Titk04]	Albayrak et al. [Alba05]	Calisti et al. [Cali04]	Sashima et al. [Sash04]	Kamara et al. [Kama04]	Lech et al. [Lech05]	Harvey et al. [Harv05]
Profil de l'utilisateur	+	+	?	?	+	+	?	?	+	+
Localisation de l'utilisateur	+	+	?	+	-	+	+	?	+	-
Préférences de l'utilisateur	-	+	+	+	+	+	+	?	+	+
Adaptation aux différents types de media	?	+	?	?	-	-	-	?	-	?
Relations entre les agents	+	?	?	?	+	?	?	+	?	+
Mécanismes de routages de requêtes	?	?	?	?	-	-	?	?	?	?
Distribution d'information entre DM	-	-	-	+	+	+	-	-	+	+

Tableau 5. Travaux basés sur des agents qui adaptent l'information dans des environnements nomades.

Plusieurs de ces sept aspects ne sont pas précisés dans les travaux. Les travaux ne détaillent pas la manière dont ils sont traités. Tel est le cas en ce qui concerne : *i)* la définition du profil de l'utilisateur ; *ii)* l'adaptation de l'affichage de différents types de média sur les dispositifs d'accès ; *iii)* la manière dont les agents sont organisés ; *iv)* la manière dont les requêtes sont traitées et redirigées vers les systèmes d'information les plus appropriés. Un aspect qui n'est pas pris en compte est la récupération de l'information provenant de *DM*. De plus, la plupart de ces travaux ignorent le fait que l'information répondant aux requêtes de l'utilisateur puisse être distribuée entre plusieurs systèmes d'information, lesquels s'exécutent sur serveurs ou sur des *DM*.

5.3. Contribution de notre proposition

La problématique abordée dans notre proposition concerne l'accès aux *SIW* et l'adaptation de l'information dans des environnements nomades. Afin de trouver des solutions à cette problématique, nous avons considéré, dans l'état de l'art, trois domaines : l'*informatique ubiquitaire*, les *systèmes multi-agents* et l'*adaptation de l'information*. De cette étude, nous avons identifié les aspects les plus importants à traiter pour résoudre cette problématique :

- L'accès aux *SIW* à travers différents dispositifs d'accès ;
- Le processus de routage de requêtes qui permet de les analyser, de sélectionner les sources d'information capables d'y répondre et de compiler les résultats ;
- La prise en compte des problèmes liés à la distribution de l'information dans plusieurs sources. Parmi ces problèmes, nous citons l'atomicité, la sécurité, la cohérence et l'intégrité ;

- La formulation des requêtes destinées à plusieurs *SIW* s'exécutant sur des serveurs ou des *DM ;*
- La définition d'une architecture pour l'organisation et gestion des agents, leurs rôles, leurs tâches, leurs communications, *etc.* ;
- Les mécanismes de représentation, de gestion et d'inférence de connaissances des agents ;
- L'affichage de l'information sur les dispositifs d'accès en considérant leurs caractéristiques et contraintes ainsi que différents types de média (texte, images, son, vidéo) ;
- L'adaptation de l'information selon le profil de l'utilisateur et les caractéristiques du dispositif d'accès ;
- La représentation des préférences de l'utilisateur correspondant aux activités que l'utilisateur souhaite accomplir dans le système, aux résultats attendus de ces activités et à la manière dont l'utilisateur désire que ces résultats soient affichés sur son dispositif d'accès ;
- La représentation du contexte d'utilisation pour la session en cours ;
- La représentation et génération du profil de l'utilisateur ;
- La prise en compte des caractéristiques contextuelles de la session en cours : la localisation de l'utilisateur, le moment de la connexion, les activités de l'utilisateur, les caractéristiques du dispositif d'accès.

Notre proposition apporte deux contributions à la problématique énoncée en considérant les aspects mentionnés ci-dessus :

Une première contribution est *PUMAS*, un *framework* construit d'une part, pour concevoir, développer et déployer des *SIWA*, et d'autre part, pour fournir à l'utilisateur une information (qui pourrait être distribuée sur des *DM* et/ou serveurs) adaptée à son profil, et aux caractéristiques de son *DM* utilisé pour la connexion. Un utilisateur peut accéder à *PUMAS* en utilisant un *DM* de type PDA, téléphone portable, *etc*. Dans *PUMAS*, chaque *DM* possède au moins un agent informant le système sur la localisation de l'utilisateur (par exemple, à l'aide d'un dispositif *GPS*) et ses caractéristiques de connexion (par exemple, le moment de connexion, le type du dispositif d'accès, le protocole de communication). Les agents de *PUMAS* peuvent, d'un côté, migrer vers différents serveurs (ou autres *DM*) où les *SIW* sont exécutés cherchant à trouver les agents pairs qui les aideront à répondre aux requêtes. D'un autre côté, ils peuvent utiliser une plate-forme centrale afin de communiquer avec d'autres agents pairs pour échanger de l'information ou pour supporter le travail coopératif entre des agents afin d'accomplir leurs tâches. A des fins d'adaptation, *PUMAS* dispose d'un *SMA d'Adaptation* (en plus des trois niveaux communs des architectures basées sur des agents, voir Tableau 3), responsable d'échanger l'information sur :

- la localisation et le moment de connexion de l'utilisateur avec les agents du *niveau des agents mobiles*,
- les caractéristiques du *DM* avec les agents du *niveau intermédiaire* et,

– les préférences de l'utilisateur avec les agents du niveau de *système d'information.*

Une deuxième contribution de notre proposition est une approche pour l'adaptation de l'information qui présente plusieurs particularités.

Premièrement, nous présentons une formalisation de la notion de *préférence utilisateur*. Cette formalisation offre un support à la représentation de trois types de préférences de l'utilisateur :

– Les *préférences d'activité* renseignent sur les *activités* qu'un utilisateur souhaite mener dans le système et sur la façon dont elles s'enchaînent (c'est-à-dire de manière séquentielle, concurrente et/ou conditionnelle).
– Les *préférences de résultat* indiquent une organisation attendue des contenus délivrés par ces activités.
– Les *préférences d'affichage* expriment comment l'utilisateur souhaite voir l'information s'afficher sur son *DM*.

Ensuite, nous prenons en compte les caractéristiques du *contexte d'utilisation* de la session en cours pour un utilisateur afin de définir son profil. Le *contexte d'utilisation* est constitué d'informations sur la localisation de l'utilisateur, les caractéristiques du *DM*, les droits d'accès de l'utilisateur et ses activités.

Par ailleurs, un algorithme de *correspondance contextuelle*, qui utilise les *préférences de l'utilisateur* et le *contexte d'utilisation*, est proposé afin de définir un *profil contextuel multidimensionnel* pour un utilisateur, lors de la session en cours. Cet algorithme analyse chaque *préférence utilisateur* afin d'évaluer si cette préférence peut être satisfaite par rapport au *contexte d'utilisation*. Ce *contexte* permet au système de sélectionner seulement les *préférences* qui sont compatibles avec les *activités* d'un utilisateur. Les préférences retenues sont les composantes du *profil contextuel de l'utilisateur*[63].

Enfin, nous avons identifié certains *conflits-types* qui peuvent survenir, par exemple, lorsque des préférences analysées comme contradictoires constituent des souhaits exprimés par l'utilisateur pour la même information. Nous avons également identifié leurs causes et la manière dont le système doit réagir afin de les résoudre, ou au moins, afin d'informer l'utilisateur sur l'existence de ces conflits.

Notre proposition ne porte pas sur la représentation du contexte d'utilisation ni sur le traitement des problèmes liés à la distribution de l'information sur plusieurs systèmes d'information. Néanmoins, notre proposition tient compte de la recherche d'information qui peut être distribuée sur plusieurs sources en définissant un mécanisme de routage de requêtes, et exploite les caractéristiques de l'informatique ubiquitaire trouvées dans le modèle de *contexte d'utilisation* proposé par Kirsch-Pinheiro [Kirs06].

[63] Le *profil contextuel de l'utilisateur* ne recouvre que des préférences de l'utilisateur.

Dans les chapitres suivants, nous présentons notre proposition : le *framework PUMAS* et les mécanismes de gestion des préférences de l'utilisateur, et de génération de son profil pour la session en cours.

PROPOSITION

6. LE FRAMEWORK PUMAS

Les caractéristiques contextuelles d'un utilisateur nomade (par exemple sa localisation, les conditions du réseau, les caractéristiques du *DM*) peuvent évoluer ce qui peut engendrer un changement des besoins d'information de l'utilisateur. A des fins d'adaptation de l'information en fonction des nouvelles caractéristiques, l'utilisation de l'approche agent est appropriée. Les agents sont des entités proactives, réactives, communicatives et capables d'accomplir de manière autonome (en fonction de leur connaissance) leurs tâches indépendamment du serveur et d'autres agents. Les agents peuvent également communiquer avec d'autres agents ou avec l'utilisateur. Les propriétés des agents leur permettent de réagir aux changements en sélectionnant les tâches les plus appropriées étant données les nouvelles caractéristiques contextuelles.

Afin de fournir aux utilisateurs nomades des facilités d'accès à des *Systèmes d'Information basés sur le Web* (*SIW*) à travers des *Dispositifs Mobiles* (*DM*), ainsi que des capacités d'adaptation des réponses délivrées, nous avons proposé *PUMAS* (acronyme de ***P****eer* ***U****biquitous* ***M****ulti-****A****gents* ***S****ystems*), un *framework* basé sur des agents ubiquitaires. Dans *PUMAS*, un agent s'exécutant sur un *DM* d'un utilisateur nomade est vu comme un *agent ubiquitaire pair*. *PUMAS* prend en compte les besoins d'information de l'utilisateur et les capacités restreintes de son *DM* pour afficher l'information. *PUMAS* traite aussi deux aspects importants d'un environnement nomade : la *localisation de l'utilisateur* (fournie, par exemple, par un dispositif *GPS*) et la distribution de l'information entre plusieurs *DM* hétérogènes (contrôlés par une approche *Pair à Pair*).

PUMAS repose également sur les trois niveaux composant l'architecture classique d'un *SIWA* (cf. section 3.2) :

- Un *niveau des agents mobiles* qui est composé du *DM* et des *agents mobiles*. L'utilisateur accède au système d'information à travers son *DM* sur lequel les *agents mobiles* s'exécutent.
- Un *niveau intermédiaire* qui offre des services (de connexion, de communication, *etc*.) afin de permettre l'interaction entre les deux autres niveaux.
- Un *niveau de système d'information* qui représente les services (en tant que réponses à des besoins fonctionnels) que le système fournit aux utilisateurs.

Dans *PUMAS*, ces trois niveaux sont représentés par des *SMA* :

- Le *SMA de connexion* (lié au *niveau des agents mobiles*) fournit le mécanisme pour faciliter la connexion des différents types de *DM* au système.
- Le *SMA de communication* (lié au *niveau intermédiaire*) assure une communication transparente entre le *DM* et le système, et affiche l'information à l'utilisateur d'une manière adaptée en tenant compte des contraintes de son *DM*.
- Le *SMA d'information* (lié au *niveau de système d'information*) reçoit la requête de l'utilisateur, la redirige vers les *Systèmes d'Information* susceptibles de répondre à la requête en prenant en compte le profil de l'utilisateur (composé par ses préférences et son histoire dans le système) ainsi que des critères tels que la proximité, la précision, le nombre de consultations, la réputation. Finalement, le *SMA d'information* retourne au *SMA de communication* des résultats adaptés à ce profil et à ces critères.

PUMAS étend l'architecture classique à trois niveaux des *SIWA* avec un nouveau niveau nommé le *Système Multi-Agents d'adaptation* (*SMA d'adaptation,* cf. section 6.1.2) dont les services et les tâches de ses agents consistent à partager et échanger (avec les agents des autres niveaux) l'information sur l'utilisateur, les caractéristiques de connexion et de communication, les caractéristiques du *DM*, *etc*.

La mobilité inhérente de l'utilisateur et des agents est supportée par des *agents ubiquitaires* pouvant être transmis à travers le réseau afin de récupérer l'information nécessaire et de la partager avec d'autres agents pour accomplir les tâches qui leur sont affectées. Nous avons choisi une *architecture P2P hybride* car ce type d'architecture permet de résoudre les problèmes suivants :

- Le contrôle de la mobilité des agents. Notamment, de telles architectures : *i)* répliquent dans le serveur les agents s'exécutant sur le dispositif d'accès, *ii)* maintiennent un agent capable de s'exécuter sur un *Système d'Information* afin d'y chercher l'information, *etc.*
- La communication entre agents en mode point à point ou en utilisant le *broadcast*. De telles architectures fournissent une infrastructure de communication entre les agents.
- La gestion de l'état de l'agent (par exemple, connecté, déconnecté, tué, *etc*.) et des services fournis par les agents. De telles architectures permettent de créer, sur la plate-forme centrale, des agents contrôlant les connexions et déconnexions des autres agents et la gestion de leurs états.

L'intégration de l'approche agent avec une architecture *P2P hybride* permet à *PUMAS* de fournir également :

- Une communication directe entre les utilisateurs et les agents ;
- l'autonomie des agents pour accomplir leurs tâches et pour jouer différents rôles;
- la manière dont les agents doivent réagir afin de répondre aux changements de l'environnement ;
- les capacités nécessaires aux agents pour exécuter leurs tâches de manière autonome ou collective en accord avec d'autres agents.

Dans *PUMAS*, un utilisateur muni d'un *DM* (par exemple, un *PDA*) peut :

- Consulter une information adaptée à différents aspects tels que le *profil de l'utilisateur* et les caractéristiques de son *contexte d'utilisation* (telles que la localisation, les activités de l'utilisateur et les caractéristiques de son dispositif d'accès) ;
- Communiquer avec d'autres utilisateurs (pairs) à travers son *DM* afin de fournir et d'obtenir une information spécifique ;
- Transmettre des requêtes vers d'autres *DM*.

Dans ce chapitre, nous présentons l'architecture logique de *PUMAS* en décrivant chaque *SMA* et nous justifions le besoin d'étendre l'architecture de base d'un *SIWA* avec un *SMA* en charge de l'adaptation. Nous présentons alors l'architecture fonctionnelle de *PUMAS*, en nous concentrant sur l'échange d'information à des fins d'adaptation et sur la représentation de connaissances.

6.1. Architecture logique

Dans *PUMAS*, les agents sont organisés en une architecture *hybride P2P* dans laquelle chaque agent peut se connecter au système ou s'en déconnecter volontairement et communiquer et/ou coopérer avec d'autres agents pairs afin d'accomplir des tâches individuelles ou collectives. Des agents qui s'exécutent sur des *DM* peuvent communiquer à travers la plate-forme centrale de *PUMAS*. Cette plate-forme s'exécute sur un serveur.

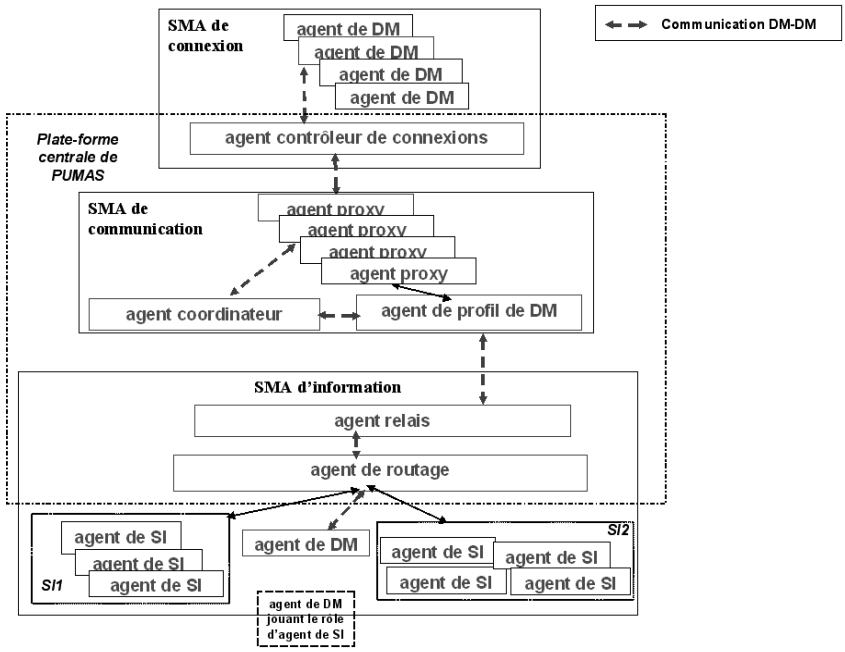


Figure 6.1. Architecture logique de PUMAS.

L'architecture de *PUMAS*, conforme à l'architecture d'un *SIWA* classique, est composée de trois *SMA* (cf. Figure 6.1) : *connexion*, *communication* et *information* où chaque agent est un *agent ubiquitaire pair*. Les agents sont connectés à une plate-forme centrale qui permet aux agents d'une part, d'enregistrer leurs services et de consulter ceux d'autres agents, et d'autre part, de communiquer avec d'autres agents. Les agents sont autonomes pour : *i)* se connecter et se déconnecter, *ii)* envoyer des messages à un agent spécifique ou à un groupe d'agents, et *iii)* accomplir leurs tâches.

Dans ce qui suit, nous décrivons les rôles des agents, les fonctionnalités de chaque composant de *PUMAS* et la manière dont les agents adaptent l'information en tenant compte des caractéristiques de l'utilisateur et celles de son *DM*.

6.1.1 L'architecture de PUMAS

L'architecture de base de *PUMAS* est composée de trois *SMA* [Carr05a] :

– Un *SMA de connexion* qui fournit les mécanismes pour faciliter la connexion de différents types de *DM* au système. Il contient un ou plusieurs *agents de DM* et un *agent contrôleur de connexions*.

– Un *SMA de communication* qui assure une communication transparente entre les *DM* et le système, et applique un *filtre d'affichage* d'information qui tient compte des caractéristiques du *DM* de l'utilisateur. Ce *SMA* contient un *agent coordinateur*, un *agent de profil de DM* et un ou plusieurs *agents proxy*.

– Un *SMA d'information* qui reçoit la requête de l'utilisateur, la redirige vers le *Système d'Information* susceptible de la traiter, applique un *filtre de contenu* en tenant compte du

profil de l'utilisateur dans le système (composé de ses préférences et de son histoire dans le système) et retourne les résultats au *SMA de communication*. Le *SMA d'information* contient plusieurs *agents relais*, plusieurs *agents de routage* et un ou plusieurs *agents de SI*.

Dans les sections suivantes, nous donnons la description des composants de *PUMAS*.

6.1.1.1 Le SMA de connexion

Il repose sur plusieurs *agents de DM* et un *agent contrôleur de connexions*. Ce dernier agent s'exécute sur la plate-forme centrale de *PUMAS* (cf. Figure 6.2).

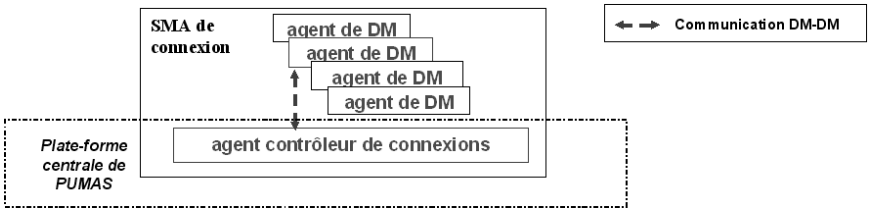


Figure 6.2. Les agents du SMA de connexion. L'agent de DM s'exécute sur le DM et l'agent contrôleur de connexions sur la plate-forme centrale de PUMAS.

Chaque *DM* peut exécuter différents *agents de DM* transmis de et vers les *SIW* (systèmes s'exécutant sur différents serveurs ou *DM*). Dans notre approche, un *agent de DM* (cf. Figure 6.3) est un *agent mobile* possédant à la fois les caractéristiques d'un *agent coopératif*[64] et celles d'un *agent de connexion*.

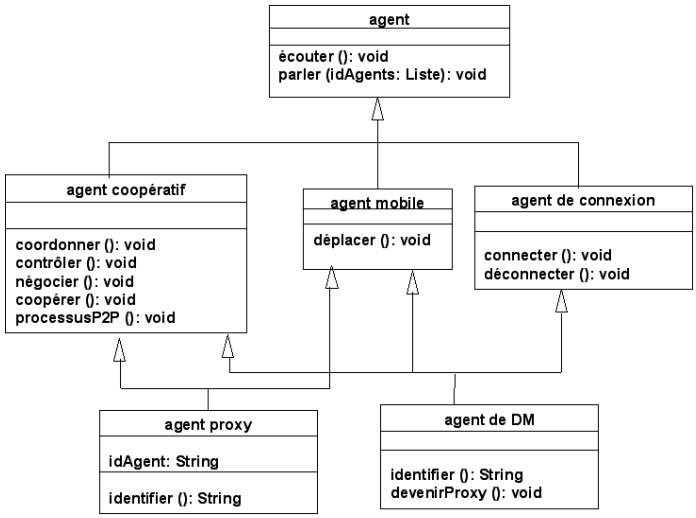


Figure 6.3. Diagramme de classes d'un agent de DM qui possède les caractéristiques et opérations d'un agent mobile, d'un agent coopératif et d'un agent de connexion.

[64] Un *agent coopératif* travaille avec d'autres agents afin d'exécuter une tache plus complexe et donc résoudre un problème également plus complexe.

L'*agent contrôleur de connexion* détecte le *type* de *DM* (par exemple, un *PDA*, un téléphone portable) en utilisant des fichiers *CC/PP* [W3Ca04] et facilite sa connexion en prenant en compte le protocole de connexion. *CC/PP* (*Composite Capability/ Preference Profiles*) est une description des capacités du dispositif d'accès et des préférences de l'utilisateur recommandée par le *W3C*. Un profil *CC/PP* peut être utilisé pour adapter l'affichage du contenu sur le dispositif. *CC/PP* utilise *RDF* (*Resource Description Framework*) pour décrire les profils. L'*agent contrôleur de connexions* sert d'intermédiaire entre le *SMA de connexion* et le *SMA de communication*. Il connaît tous les agents dans le système à travers un mécanisme de pages jaunes : les informations concernant les agents proposent une description de leurs connexions, leurs états, les services fournis, leur localisation.

Afin d'implémenter le *SMA de connexion*, il est nécessaire de définir les niveaux de communication et d'infrastructure entre les *agents de DM* et l'*agent contrôleur de connexions* pour rendre transparente la connexion des différents types de *DM* et pour être en mesure d'afficher l'information (réponses aux requêtes d'information) en considérant les contraintes matérielles et logicielles du *DM*. Ce *SMA* doit donc disposer d'un modèle du *DM* en charge de stocker l'information sur des caractéristiques techniques spécifiques. Ce modèle sera exploité afin d'adapter la présentation de l'information aux contraintes physiques du *DM*.

6.1.1.2 Le SMA de communication

Ce *SMA* offre une interface qui rend transparente la communication entre les utilisateurs et active les mécanismes pour afficher l'information d'une manière adaptée au *DM*. Il repose sur plusieurs *agents proxy*, sur un *agent de profil de DM* et sur un *agent coordinateur*. Tous ces agents s'exécutent sur la plate-forme centrale de *PUMAS* (comme le montre la Figure 6.4).

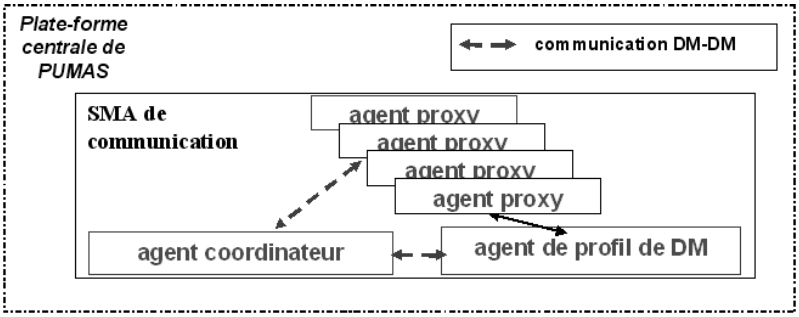


Figure 6.4. Communication entre les agents du SMA de communication.

Un *agent proxy* est une représentation d'un *agent de DM* : il possède les mêmes propriétés et comportements qu'un *agent de DM* excepté celles relatives à la connexion (cf. Figure 6.3). Celles-ci sont inutiles car l'*agent proxy* s'exécute sur la plate-forme centrale (cf. Figure 6.4). Un *agent proxy* représente une connexion d'un *agent de DM*. Deux utilisateurs différents peuvent se connecter au système à travers le même *DM*, chacun étant représenté par un *agent de DM*. Chaque *agent de DM* sera associé à un *agent proxy* différent. On distingue donc, dans ce cas, deux agents : un *agent de DM* sur le *DM* et un *agent proxy* sur la plate-forme centrale de *PUMAS*.

Un *agent de DM* (appartenant au *SMA de connexion*) possède les caractéristiques d'un *agent coopératif* et *mobile*. Il peut également jouer lui-même le rôle d'*agent proxy*. Dans ce cas, il y a seulement un agent dans le système - l'*agent de DM* - qui peut être transmis au système (ou à un autre *DM*), être exécuté sur le serveur (ou sur un autre *DM*), et jouer le rôle d'un *agent proxy*.

Dans *PUMAS*, l'accent est mis sur la représentation de connaissances et du comportement tant de l'*agent proxy* que de l'*agent de DM*, tous deux considérés comme des *agents coopératifs* (voir Figure 6.3). A ce titre, l'*agent proxy* et l'*agent de DM* sont capables d'accomplir les tâches de *Coordination*, *Coopération*, *Contrôle* et *Négociation* (*CCCN*) dans le *SMA* (cf. section 3.2). Egalement, ils adoptent le *processus P2P* décrit par Panti *et al.* [Pant02] (cf. section 3.2) afin de définir la stratégie à appliquer pour accomplir les tâches assignées (de type *CCCN* ou autre) et pour découvrir les pairs capables de coopérer et de contribuer avec eux à l'exécution de tâches plus complexes.

L'*agent de profil de DM* regroupe les caractéristiques du *DM*, parmi lesquelles on compte divers aspects tels que les contraintes techniques et de connexion du *DM*, son comportement dans certains types de réseau, *etc*. De plus, cet agent, à l'aide de l'*agent coordinateur*, établit les mécanismes pour échanger des données hypermédias avec l'utilisateur (*filtre d'affichage*, cf. section 8.6). Ainsi, si le résultat d'une requête de l'utilisateur contient plusieurs images, ces agents définissent l'ordre et le nombre d'images à afficher sur l'écran en fonction des capacités du *DM* de l'utilisateur.

L'*agent coordinateur* est en communication permanente avec l'*agent contrôleur de connexions* (appartenant au *SMA de connexion*) afin de vérifier, grâce aux pages jaunes (qui contiennent la liste d'agents, leurs services et leur état), l'état de la connexion des *agents de DM*. S'il y a des problèmes avec l'*agent contrôleur de connexions* (par exemple, si l'*agent contrôleur de connexions* échoue ou s'il n'est pas capable de gérer toutes les connexions car elles sont nombreuses), l'*agent coordinateur* peut jouer le rôle d'*agent contrôleur de connexions* jusqu'à la résolution des problèmes survenus (cf. section 6.2.1.2).

6.1.1.3 Le SMA d'information

Le *SMA d'information* est composé de plusieurs *agents relais*, de plusieurs *agents de routage*, et d'un ou plusieurs *agents de SI*. Comme le montre la Figure 6.5, hormis les *agents de SI* qui s'exécutent sur les dispositifs hébergeant les *Systèmes d'Information*, tous ces agents s'exécutent sur la plate-forme centrale de *PUMAS*.

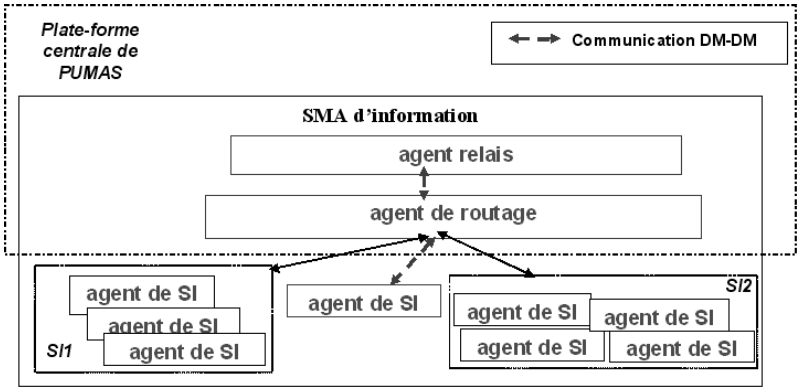


Figure 6.5. Communication entre les agents du SMA d'information.

La structure interne du *SMA d'information* est composée : d'agents associés aux différents *SIW* (qui peuvent être de type *SMA* ou non), d'*agents de routage* qui redirigent les requêtes de l'utilisateur vers les *SI* les plus pertinents (*processus de routage de requêtes*, section 7), d'*agents relais* qui définissent les profils des utilisateurs à partir de leurs préférences ou de leur historique dans le système, et possèdent une vue générale de tout le système. Les *agents relais* connaissent les agents des *SMA de communication* et *d'information*, leurs services, leurs localisations, leurs profils. Ils possèdent aussi une vue générale des agents de *PUMAS*. Un *agent relais* reçoit des requêtes transmises du *SMA de communication* et les redirige aux *agents de routage*. Ces *agents relais* ont pour rôle de vérifier si les résultats de la requête sont conformes au profil de l'utilisateur.

Afin de rediriger la requête vers un ou des *Systèmes d'Information*, un *agent de routage* adopte une stratégie reposant sur plusieurs critères (la localisation de l'utilisateur, les activités des pairs, les contraintes du *DM*, les préférences de l'utilisateur, *etc*.). La stratégie consiste en l'envoi de la requête à un seul *Système d'Information* (ou à plusieurs simultanément), la décomposition de la requête en sous-requêtes (chacune étant envoyée à un ou plusieurs *Systèmes d'Information*). L'*agent de routage* est également chargé de compiler les résultats qu'il obtient des *Systèmes d'Information* et de les analyser afin de décider ce qui doit être retourné à un *agent relais*.

Un *agent de SI* reçoit la requête de l'utilisateur de l'*agent de routage*. Il est en charge de la traiter en cherchant des éléments de réponse dans le *Système d'Information*. Une fois que le résultat de la requête obtenu, l'*agent de SI* le renvoie à l'*agent de routage*. Un *agent de SI* peut répondre à la requête lui-même ou déléguer cette tâche au composant approprié du *Système d'Information*. Cela dépend notamment de la nature du *SIW*. Notre approche recouvre des *Systèmes d'Information* complexes et éventuellement distribués, exécutés sur des serveurs, mais également des *Systèmes d'Information* très simples, seulement construits sur quelques fichiers et complètement hébergés et exécutés sur des *DM*. Dans ce dernier cas, un *agent de SI* peut être suffisant pour assurer le fonctionnement correct du *SMA d'information*. Il est important de noter que, dans ce cas, ce que nous appelons un « *agent de SI* » est en fait l'*agent de DM* d'un *DM* jouant le rôle d'un *agent de SI* puisqu'il possède la connaissance requise (sur les fichiers stockés

dans le *DM*) pour exécuter une requête. Dans un *Système d'Information* complexe, l'*agent de SI* peut collaborer avec d'autres *agents de SI* (si le *Système d'Information* a été développé suivant le paradigme *SMA*) ou avec un autre composant du *Système d'Information* pour exécuter la requête. Dans le cas d'un *Système d'Information* non basé sur des *SMA*, notre approche requiert seulement qu'un *agent de SI* soit développé afin d'assurer les communications entre *PUMAS* et le *Système d'Information*.

La localisation du *Système d'Information* pourrait changer, notamment si ce *Système d'Information* s'exécute sur un *DM*. L'*agent de SI*, qui s'exécute sur ce *Système d'Information*, informe des changements de localisation à l'*agent de routage*. Un autre aspect à notifier est la modification de l'information gérée par le *Système d'Information*.

A des fins d'adaptation, nous avons intégré dans *PUMAS* un quatrième *SMA* en charge d'adapter l'information en considérant le profil de l'utilisateur, les caractéristiques techniques de son *DM* et des caractéristiques contextuelles. Ce nouveau *SMA* est présenté dans la section suivante.

6.1.2 Le SMA d'adaptation

Les capacités d'adaptation de *PUMAS* reposent sur un processus de filtrage à deux étapes. Premièrement, un *filtre de contenu* permet de sélectionner l'information la plus pertinente en prenant en compte le profil de l'utilisateur. Deuxièmement, *le filtre d'affichage* est appliqué aux résultats du premier filtre et tient compte des caractéristiques et des contraintes techniques du *DM* de l'utilisateur (cf. section 8.6).

Les rôles des agents des *SMA* de *PUMAS* décrits jusqu'ici, ne recouvrent pas de tâches telles que :

- la gestion de préférences de l'utilisateur pour une session spécifique à partir d'un ou plusieurs dispositifs d'accès ;
- la gestion de l'historique de préférences de l'utilisateur ;
- le regroupement de caractéristiques d'un *DM* ;
- le regroupement de problèmes détectés pendant les connexions d'un type de *DM*.

Ces tâches sont néanmoins nécessaires pour adapter l'information. C'est pourquoi, nous avons étendu *PUMAS* avec un quatrième *SMA* appelé le *SMA d'adaptation*, composé également d'*agents ubiquitaires* [Carr05b]. Les services et les tâches de ces agents consistent essentiellement à gérer des fichiers écrits en *OWL* contenant de l'information sur l'utilisateur et son *DM*. Les agents du *SMA d'adaptation* possèdent aussi de la connaissance (stockée dans une *base de connaissances*), relative à l'utilisateur, permettant de sélectionner et de filtrer l'information pour les utilisateurs. Cette connaissance est acquise en analysant l'historique de l'utilisateur dans le système : ses dernières connexions, requêtes, préférences, *etc*. Les agents du *SMA d'adaptation* communiquent avec les agents des *SMA de connexion, de communication* et *d'information* afin de partager et d'échanger des informations sur l'utilisateur (explicitement

extraites des fichiers écrits en *OWL* ou inférées à partir de leurs règles et connaissances), les caractéristiques de connexion et de communication, les caractéristiques du *DM*, *etc*.

Le *SMA d'adaptation* est composé de plusieurs *agents d'utilisateur*, d'un *agent de filtre d'affichage* et d'un *agent de filtre de contenu*. Ces agents s'exécutent sur la plate-forme centrale de *PUMAS* (cf. Figure 6.6).

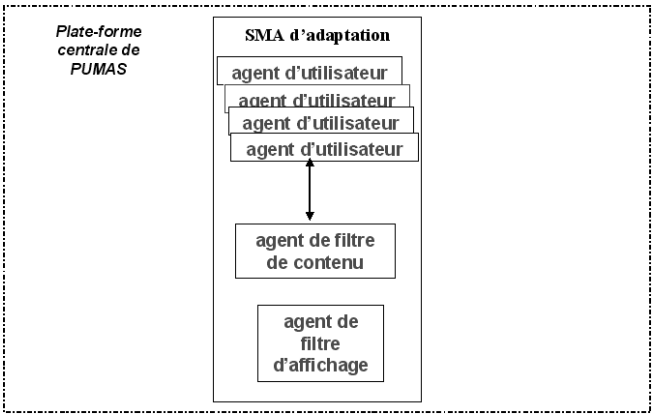


Figure 6.6. Le SMA d'adaptation.

Chaque *agent d'utilisateur* gère l'information concernant les caractéristiques personnelles de l'utilisateur (son identité, sa localisation, *etc*.) et ses préférences (en termes de présentation de l'information). Un *agent d'utilisateur* représente un utilisateur. L'*agent d'utilisateur* communique avec les *agents* de *DM* et les *agents proxy* (qui appartiennent respectivement au *SMA* de *connexion* et *de communication*) pour analyser et centraliser toutes les caractéristiques du même utilisateur car celui-ci peut accéder au système à travers plusieurs *DM*. Ces agents échangent les informations concernant les caractéristiques de l'utilisateur et celles de la session en cours.

L'*agent de filtre d'affichage* gère l'information sur les caractéristiques de différents types de *DM* (par exemple, les formats de fichiers supportés) et la connaissance acquise au cours des connexions précédentes (par exemple, les problèmes de connectivité rencontrés avec certains types de réseaux ou dans certains endroits).

L'*agent de filtre de contenu* gère l'information sur les préférences et l'historique de l'utilisateur dans le système. Cet *agent* communique les *préférences de l'utilisateur* à l'*agent relais* (appartenant au *SMA d'information*) qui les ajoute aux requêtes de l'utilisateur et vérifie si les résultats sont adaptés en tenant compte de cette information.

En toute généralité, les agents du *SMA de connexion, de communication* et *d'information* communiquent avec les agents du *SMA d'adaptation* afin de compléter les requêtes de l'utilisateur par des informations destinés à l'adaptation, puis, de vérifier les résultats fournis par l'*agent de routage* appartenant au *SMA d'information* (cf. Figure 6.7).

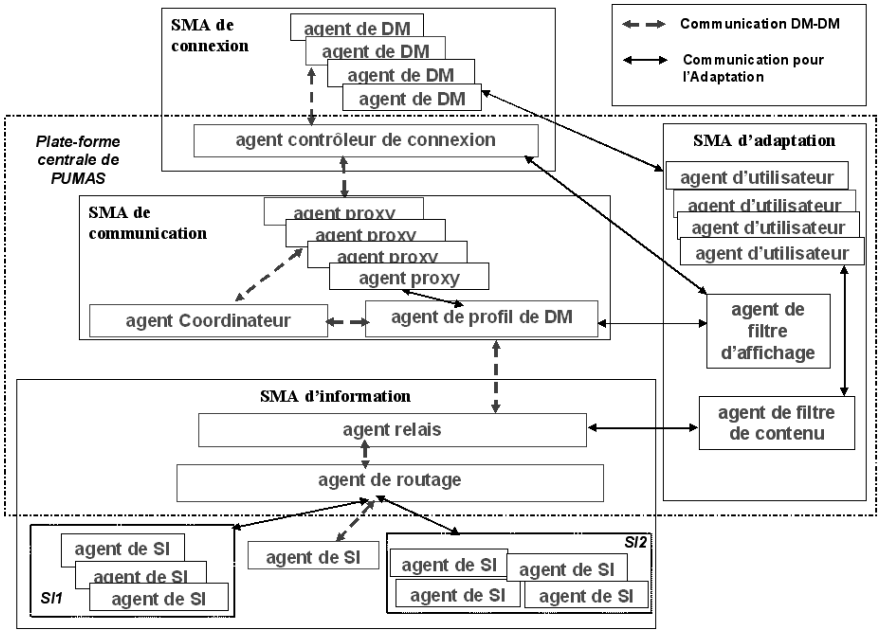


Figure 6.7. Architecture logique de PUMAS contenant le SMA d'adaptation.

6.1.3 Définition de profils et de règles pour les agents

Afin de définir les profils des utilisateurs, il faut disposer d'un mécanisme qui détecte les utilisateurs et leurs activités dans le système, d'une représentation des critères utilisés pour définir le profil (par exemple, les préférences de l'utilisateur, son historique dans le système, les caractéristiques contextuelles de la session en cours *etc*., cf. section 8.2), et d'un algorithme qui exploite cette représentation pour la génération du profil de l'utilisateur [Nage99]. A des fins de représentation de l'information prise en compte pour la définition du profil de l'utilisateur, nous reprenons les extensions de *CC/PP* proposées par Indulska *et al.* [Indu03a] qui définissent des profils généraux de *DM*. En vue de la définition des comportements et des activités des agents, nous exploitons des ontologies en utilisant le langage *OWL* (*Ontology Web Langage*, cf. section 3.4.1.2) qui s'impose comme le standard de descriptions d'ontologies, recommandé par le *W3C*. *OWL* peut être utilisé pour décrire un *SMA* comme un ensemble de classes (agents) et de relations (communications et représentation de connaissances) entre eux en utilisant des fichiers écrits en *OWL*. Les composants *OWL* sont des classes, attributs, instances de classes et relations de classes. La connaissance des agents de *PUMAS* est définie en utilisant *JESS* (cf. section 6.2.4.2).

6.2. Architecture fonctionnelle

Nous présentons ici l'architecture fonctionnelle de *PUMAS* et nous décrivons les données échangées et les communications entre les agents permettant d'adapter l'information à l'utilisateur.

6.2.1 Echange d'information pour l'adaptation

D'après Kothari [Koth97], un agent doit stocker et gérer des métadonnées sur l'information à partager avec d'autres agents, l'information sur son état et son contexte. En accord avec ce principe, *PUMAS* est développé de telle sorte que le serveur stocke ses propres données (connaissances échangées entre les agents de *PUMAS*), ses règles et son contexte. Chaque agent doit quant à lui stocker ses connaissances (propres, acquises, du contexte), ses services (ce qu'il peut faire, ce qu'il a appris à faire), et les rôles qu'il peut jouer dans le système. Finalement, chaque agent doit stocker son cycle de vie (c'est-à-dire, des informations sur le moment où il a été créé, les tâches qu'il a exécutées lors d'une session, et le cas échéant, le moment où il a été détruit). Dans cette section, nous présentons l'information échangée par les agents de *PUMAS* en vue de l'adaptation. Ces échanges reposent sur les relations entre agents qui sont également décrites. Les descriptions de la session de l'utilisateur, de son profil, des caractéristiques de son dispositif d'accès et de sa localisation sont contenues dans des fichiers écrits en *OWL*, et données dans l'annexe 1[65].

6.2.1.1 Echanges impliquant les agents du SMA de connexion

La connaissance de l'*agent de DM* concerne les caractéristiques du type de *DM* utilisé (par exemple, un *PDA*). Les activités de cet agent sont définies en fonction de l'application (par exemple, cet agent est utilisé pour transmettre un fichier). L'*agent de DM* gère un fichier *de dispositif* (écrit en *OWL* et stocké sur le *DM* de l'utilisateur) qui décrit les caractéristiques du *DM* en accord avec une ontologie commune pour tous les agents partageant cette information. L'*agent de DM* envoie ce fichier à l'*agent de filtre d'affichage* via l'*agent contrôleur de connexions* lors de la connexion de l'utilisateur. Ce fichier contient des informations sur les besoins de l'application, l'état du réseau, les formats de fichiers hypermédias supportés par le *DM*, les conditions de déconnexion (automatique après un intervalle de temps coulée ou volontaire).

Un *agent de DM* gère également un fichier de *session* décrivant, en *OWL*, les caractéristiques de la session de l'utilisateur : l'identification (*ID* de l'utilisateur), le temps de début de la session (*TempsDebut*), le type du *DM* connecté (*DMCourant*) et les activités courantes de l'utilisateur (avec leur état). Lors de la connexion de l'utilisateur, ce fichier est envoyé à l'*agent d'utilisateur* (appartenant au *SMA d'Adaptation*) et celui-ci est en charge de le modifier dans le cas où un changement survient (par exemple, le début ou la terminaison d'une activité). L'information dans ce fichier permet à l'utilisateur (à travers son *agent de DM*) de connaître l'état de ses activités à tout moment, en particulier après une déconnexion involontaire. Cette information est envoyée dans le sens inverse, c'est-à-dire de l'*agent d'utilisateur* vers l'*agent de DM*.

L'*agent contrôleur de connexions* s'exécute sur la plate-forme centrale de *PUMAS*, et obtient la localisation de l'utilisateur et le type de *DM* de l'utilisateur respectivement par

[65] http://www-lsr.imag.fr/users/Angela.Carrillo/

l'intermédiaire du fichier *de localisation* et du fichier *de dispositif*. Les deux fichiers (écrits en *OWL*) sont fournis par les *agents de DM* et localement gérés (sur la plate-forme centrale) par l'*agent contrôleur de connexions*. Cet agent complète toutes les requêtes reçues de l'*agent de DM* avec cette information (cf. section 7.3).

Cet agent vérifie aussi les connexions établies par les utilisateurs à travers leurs *DM* et les états des agents (par exemple, connecté, déconnecté, tué, *etc*.), et lie chaque *agent de DM* avec son correspondant *agent proxy* dans le *SMA de communication* (cf. section 6.1.1.2). L'*agent contrôleur de connexions* introduit par *PUMAS*, permet de vérifier si l'utilisateur est encore connecté. Si non, il vérifie s'il s'est volontairement déconnecté ou si la déconnexion a été produite par un défaut (par exemple, un problème du système ou de réseau). L'*agent contrôleur de connexions* vérifie si l'utilisateur souhaite poursuivre sa session (il serait alors représenté par le même *agent proxy*) ou s'il souhaite ouvrir une nouvelle session (il serait dans ce cas représenté par un nouvel *agent proxy*).

6.2.1.2 Echanges impliquant des agents du SMA de communication

L'*agent de profil de DM* partage l'information sur les caractéristiques d'un *DM* spécifique avec l'*agent de filtre d'affichage* (appartenant au *SMA d'adaptation*) afin d'augmenter une requête avec ces caractéristiques, lors de la première étape du *filtre d'affichage*. Lorsque l'*agent de profil de DM* reçoit la requête de l'*agent coordinateur*, il demande à l'*agent de filtre d'affichage* les caractéristiques du *DM* et ajoute cette information à la requête. Cependant, les caractéristiques du *DM* peuvent changer pendant l'exécution de la requête (par exemple, l'utilisateur change de *DM* ou les conditions du réseau changent), l'*agent de profil de DM* est notifié de ces changements par l'*agent contrôleur de connexions* (via l'*agent de filtre d'affichage* qui lui fournit les nouvelles caractéristiques du *DM*) et vérifie que les résultats de la requête (provenant de l'*agent relais*) sont toujours en adéquation avec les changements survenus. Une fois les résultats vérifiés et éventuellement recalculés (si des modifications ont été observées), l'*agent de profil de DM* les envoie à l'*agent coordinateur*.

Le recalcul des résultats peut être contrôlé de deux manières : *i)* en informant l'utilisateur sur les changements survenus lors du traitement des requêtes et en lui demandant si ces changements sont pertinents au regard des nouvelles caractéristiques contextuelles, *ii)* en recalculant les résultats après un nombre fixe d'itérations défini par l'utilisateur ou par le concepteur de l'application. Pour chaque itération, l'application peut informer l'utilisateur sur les changements détectés et peut lui demander si le recalcul de résultats est nécessaire. Une réponse défavorable de la part de l'utilisateur finit le recalcul, suivi de l'envoi des résultats à l'utilisateur.

Dans le cas de problèmes sur l'*agent contrôleur de connexions*, l'*agent coordinateur* peut jouer le rôle du premier. Une fois les problèmes surmontés par l'*agent contrôleur de connexions*, les deux agents doivent synchroniser l'information à propos des agents connectés. Pour cette synchronisation, nous proposons la gestion de trois listes : une liste des agents connectés à *PUMAS*, une liste des agents déconnectés (gérées par l'*agent contrôleur de connexions*), ainsi qu'une liste avec tous les agents qui se sont connectés (historique) ; la

description de chacun de ces agents est accompagnée de son état courant. Cette dernière liste est gérée par l'*agent coordinateur*. Chaque fois qu'un agent se connecte, l'*agent contrôleur de connexions* ajoute l'identifiant de cet agent à la liste des agents connectés et notifie cette connexion à l'*agent coordinateur* qui à son tour l'ajoute à l'historique avec l'état « *connecté* ». De la même manière, lorsqu'un agent se déconnecte, l'*agent contrôleur de connexions* l'ajoute à la liste des agents déconnectés et l'élimine de la liste des agents connectés. Il notifie cette déconnexion à l'*agent coordinateur* qui l'ajoute ensuite à l'historique avec l'état « *déconnecté* ». L'*agent contrôleur de connexions* envoie périodiquement un message à l'*agent coordinateur* afin de l'informer qu'il est encore actif. Dans le cas où l'*agent coordinateur* ne reçoit pas ce message comme prévu, l'*agent coordinateur* prend la place de l'*agent contrôleur de connexions*, en mettant à jour l'état des agents qui se connectent ou se déconnectent. Une fois que l'*agent coordinateur* est informé par l'*agent contrôleur* de son rétablissement, l'*agent coordinateur* lui envoie un message avec les deux listes, celle des agents connectés et celle des agents déconnectés, gérées pendant la période où il n'y a pas eu de communication entre ces deux agents. L'*agent contrôleur de connexions* met à jour ses listes avec l'information reçue.

6.2.1.3 Les agents du SMA d'adaptation

Chaque *agent d'utilisateur* gère un fichier *d'utilisateur* (écrit en *OWL*) qui contient les caractéristiques personnelles d'un utilisateur (par exemple, *ID* de l'utilisateur dans le système, localisation, *etc*.) et ses préférences (cf. (1) dans Figure 6.8). Ce fichier est obtenu à travers les *agents de DM* (cf. (2) dans Figure 6.8). L'*agent d'utilisateur* communique avec l'*agent de filtre de contenu* afin d'envoyer le fichier *d'utilisateur* pour mettre à jour les préférences de l'utilisateur (cf. (3) dans Figure 6.8).

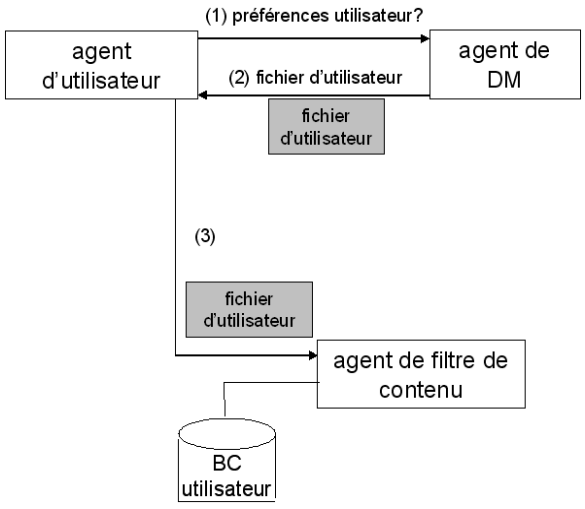


Figure 6.8. Echange d'information de l'agent d'utilisateur.

L'*agent de filtre d'affichage* gère une *base de connaissances* contenant l'information générale sur les caractéristiques de différents types de *DM* (par exemple, les formats de fichiers supportés) et la connaissance acquise au cours des connexions précédentes (par exemple, les

problèmes et capacités du réseau par rapport à la transmission de données). L'*agent de filtre d'affichage* connaît le type de dispositif à travers lequel l'utilisateur est connecté en utilisant le fichier *de dispositif* (écrit en *OWL* et fourni une fois pour toutes par l'*agent contrôleur de connexions* appartenant au *SMA de connexion*, cf. (1) et (2) de la Figure 6.9). Si lors d'une session, l'utilisateur change de *DM*, l'*agent de filtre d'affichage* reçoit un nouveau fichier *de dispositif* de l'*agent de DM* (via l'*agent contrôleur de connexions*). L'*agent de filtre d'affichage* échange l'information sur le *DM* et l'information qu'il possède dans sa base de connaissances sur ce type de dispositif avec l'*agent de profil de DM* (cf. (4) Figure 6.9) qui regroupe les caractéristiques, les problèmes et les contraintes techniques du *DM* de l'utilisateur (cf. (3) Figure 6.9). Il est important de noter que cette information sera ajoutée à la requête de l'utilisateur afin d'adapter les résultats en tenant compte des caractéristiques du *DM* (cf. section 7.3.2).

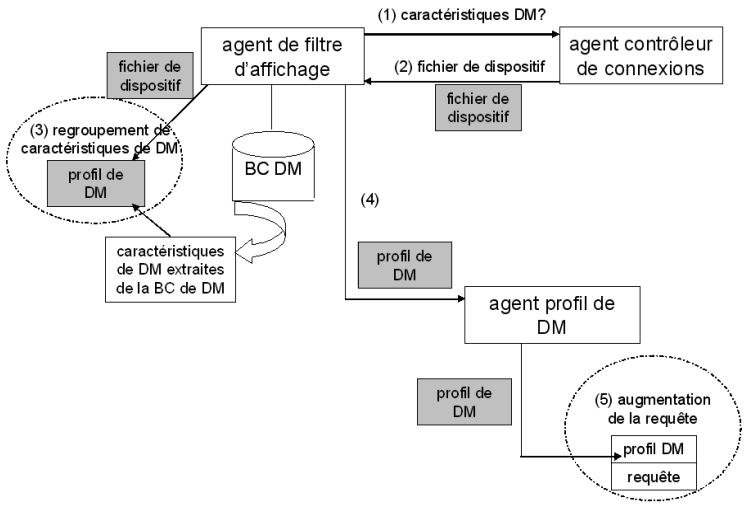


Figure 6.9. Echange d'information de l'agent de filtre d'affichage.

L'*agent de filtre de contenu* gère une *base de connaissances* contenant les préférences et les caractéristiques de l'utilisateur afin de garder son histoire dans le système. Il communique avec l'*agent d'utilisateur*, en demandant les préférences de l'utilisateur définies pour la session en cours (cf. (1) dans Figure 6.10). L'*agent d'utilisateur* lui envoie le fichier *d'utilisateur* (cf. (2) dans Figure 6.10) et l'*agent de filtre de contenu* crée le profil de l'utilisateur en se basant sur ce fichier et sur les profils générés pour les sessions précédentes de cet utilisateur (cf. (3) dans Figure 6.10). L'*agent de filtre de contenu* ne reçoit qu'une fois le fichier *d'utilisateur* correspondant à la session demandée, mais si lors d'une session, l'utilisateur change de préférences, l'*agent de filtre de contenu* reçoit un nouveau fichier *d'utilisateur* de l'*agent de DM* (via l'*agent contrôleur de connexions*). Afin de créer le profil de l'utilisateur pour la session en cours, cet agent utilise les préférences de l'utilisateur contenues dans ce fichier (cf. section 8.2.2). L'information sur le profil de l'utilisateur est échangée avec un *agent relais* (cf. (4) dans Figure 6.10) et sera ajoutée à la requête de l'utilisateur afin d'adapter les résultats aux caractéristiques de l'utilisateur (cf. section 7.3.2).

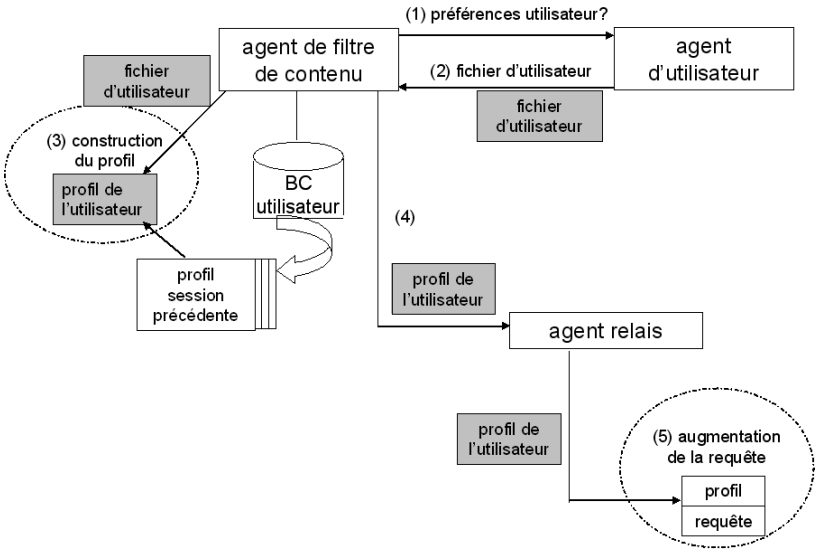


Figure 6.10. Echange d'information de l'agent de filtre de contenu.

6.2.2 Envoi des requêtes de l'utilisateur

PUMAS a la capacité de répondre à des requêtes de l'utilisateur sans préciser un *Système d'Information* spécifique. Les principaux agents impliqués dans ce processus sont les *agents relais*, les *agents de routage* et les différents *agents de SI*, chacun s'exécutant sur un *Système d'Information*. Ce processus est présenté dans la section 7. Nous présentons ici, le passage de messages contenant des requêtes de l'utilisateur entre les agents du *SMA de connexion, de communication* et l'*agent relais* (appartenant au *SMA d'information*). Nous décrivons également les messages contenant les résultats de ces requêtes. De tels messages suivent le chemin inverse (de l'*agent relais* vers l'*agent de DM*).

La Figure 6.11 présente la manière dont une requête d'information est propagée de l'*agent de DM* vers un *agent relais*. Cette figure est un diagramme de séquence *AUML* représentant les interactions (messages) entre les agents. Ces messages sont appelés *actes de communication* (par exemple, « *accept* », « *inform* », « *propose* », « *query* », « *subscribe* », « *propagate* », cf. section 3.3.2). Les messages peuvent être envoyés d'une manière concurrente ou comme résultat de l'évaluation de certaines conditions. La Figure 6.11 présente la manière dont un *agent de DM* envoie une requête à l'*agent contrôleur de connexions*. Cet *agent contrôleur de connexions* peut envoyer un message afin de créer un *agent proxy* (si un *agent proxy* n'existe pas dans le système pour représenter l'*agent de DM*). L'*agent contrôleur de connexions* peut ensuite envoyer un message afin d'informer l'*agent proxy* que l'*agent de DM* correspondant a besoin d'information. L'*agent relais* (appartenant au *SMA d'information*) reçoit, d'une part, les requêtes en leur ajoutant l'information sur les *préférences de l'utilisateur*, et d'autre part, leurs résultats en vérifiant s'ils sont en accord à ces préférences. L'*agent de profil de DM* (appartenant au *SMA d'information*) reçoit, d'une part, les requêtes en leur ajoutant l'information sur les caractéristiques du dispositif d'accès, et d'autre part, leurs résultats en vérifiant s'ils sont en accord avec ces caractéristiques. L'*agent relais* envoi les résultats des

requêtes à l'*agent de profil de DM* dans un message qui peut être de trois types : *i)* refus (« *refuse* ») pour refuser les résultats de la requête car le délai d'attente est dépassé (« *timeout* ») ; *ii)* incompréhension (« *not understand* ») pour ne pas accepter les résultats de la requête car ils ne sont pas conformes aux préférences de l'utilisateur ; *iii)* proposition (« *propose* ») pour lui proposer les résultats de la requête.

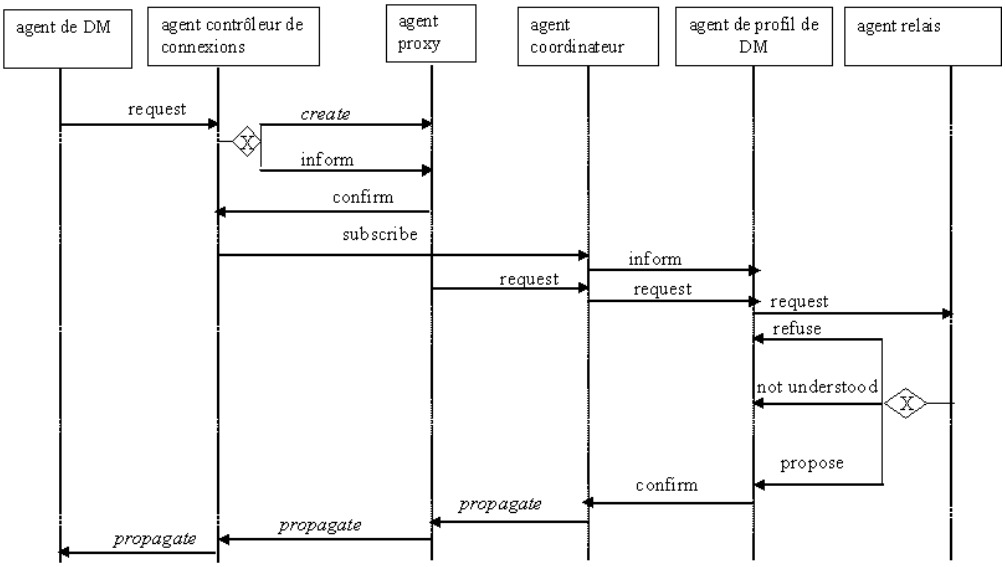


Figure 6.11. Diagramme de séquence pour envoyer une requête d'information.

Afin d'illustrer les rôles des agents de *PUMAS*, nous présentons un exemple d'applications destinées à des médecins [Carr04]. L'objectif de l'utilisation de *PUMAS* pour le médecin est de gérer son emploi du temps et de fixer les dates des opérations chirurgicales dans lesquelles interviennent ses collègues.

6.2.3 Un exemple d'utilisation de PUMAS

Dans cette section, nous présentons un cas d'application de *PUMAS*. Ce cas montre le chemin suivi par les requêtes de l'utilisateur dans un contexte particulier. L'utilisateur de *PUMAS* est un médecin chirurgien exerçant dans plusieurs hôpitaux et dans son cabinet privé. Ce médecin possède différents types de *DM* (par exemple, PDA, téléphone portable, ordinateur portable, *etc*.). Chaque jour, il souhaite consulter son emploi du temps afin de connaître ses activités et de programmer les *rendez-vous* avec ses patients et les opérations chirurgicales dans lesquelles il intervient. S'il est à son cabinet, il souhaite interroger son emploi du temps : cela correspond à une requête sur ses *rendez-vous* avec ses patients. S'il est dans un hôpital, en revanche, la requête sur son emploi du temps doit l'informer des opérations qu'il doit effectuer dans cet hôpital. Il peut également exécuter d'autres services comme la consultation de son emploi du temps, sans que n'intervienne sa localisation actuelle, en fournissant comme paramètre le cabinet ou l'hôpital pour lequel il voudrait connaître son emploi du temps. De la même manière, il peut communiquer avec d'autres collègues afin de fixer la date d'une

opération chirurgicale, ou il peut consulter le système qui gère l'emploi du temps des médecins d'un certain hôpital.

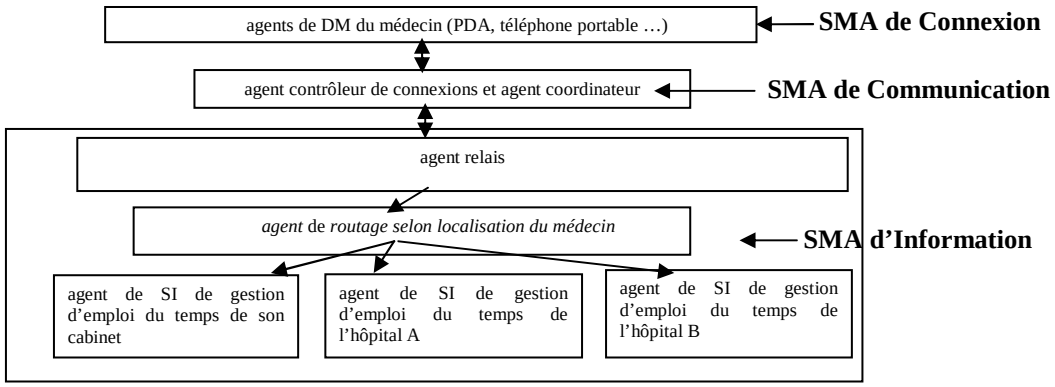


Figure 6.12. Un système d'emploi du temps géré par PUMAS.

Le médecin peut demander des informations sur les emplois du temps d'autres médecins intervenant sur les mêmes opérations chirurgicales dans un hôpital. Lorsqu'un médecin a besoin d'information, l'*agent de DM* s'exécutant sur son *DM* se connecte au système à travers les *agents contrôleur de connexions* et *coordinateur* contrôlant les différentes connexions (cf. Figure 6.12). Dans un premier temps, l'*agent de DM* demande directement cette information aux autres agents s'exécutant sur les *DM* des médecins potentiellement concernés, avec lesquels le médecin opère fréquemment dans cet hôpital. Si l'information est obtenue, les *agents de DM* l'échangent. Une autre solution consiste à ce que le médecin communique à travers son *DM* avec le système qui gère son emploi du temps (l'application) en utilisant les agents du *SMA de communication* (les *agents contrôleur de connexions* et *coordinateur*). Ces agents vérifient le type de son *DM* et passent la requête au *SMA d'information*, plus précisément à l'*agent relais* qui analyse les préférences de l'utilisateur les plus pertinentes pour la requête (par exemple, la localisation, son emploi du temps, *etc*.). En utilisant la localisation du médecin, l'*agent de routage* identifie le cabinet ou l'hôpital où le médecin se trouve, puis, envoie la requête au *Système d'Information* gérant l'agenda des médecins chirurgiens concernés (dans cet exemple, le *Système d'Information* du cabinet du médecin ou le *Système d'Information* de l'hôpital dans lequel il a passé la requête, voir Figure 6.12). Lorsque l'*agent relais* obtient les résultats de l'*agent de routage* (il n'obtient que l'information du système qui gère l'emploi du temps de l'hôpital où le médecin intervient ou de son cabinet), l'*agent relais* les renvoie à son tour aux agents du *SMA* de *communication* qui vérifient le type de cette information (par exemple, image, audio, vidéo) et, à l'aide de l'*agent contrôleur de connexions*, définit la manière de l'afficher au médecin en considérant les caractéristiques de son *DM*. Finalement les résultats de la requête son envoyés à l'agent de *DM* du médecin pour les afficher sur le *DM* du médecin.

6.2.4 Représentation de connaissances

6.2.4.1 Les ontologies

Nous avons choisi les ontologies comme mécanisme de représentation de connaissances (cf. section 3.4.1). D'après Pan *et al.* [PanC03], une *ontologie* fournit une compréhension partagée de la connaissance d'un domaine permettant la communication des agents dans un

environnement ouvert inter-operable. Ces ontologies expriment donc une représentation commune de l'information partagée par les agents. *PUMAS* définit à des fins d'adaptation quatre ontologies[66] qui expriment : *i)* les caractéristiques de la session de l'utilisateur ; *ii)* celles de son dispositif d'accès ; *iii)* celles de ses préférences ; *iv)* celles de sa localisation. Les concepts appartenant à chacune de ces ontologies sont inspirés des travaux d'Indulska *et al.* [Indu03a].

La première ontologie décrit les concepts liés aux caractéristiques d'un *DM* : les besoins de l'application (matériels, logiciels, de navigation, de qualité de service) et les caractéristiques du réseau (l'état de déconnexion et l'interface de réseau). La configuration attendue par une application regroupe les informations liées : au système d'exploitation, au profil MIDP-PJava pour le développement des applications sur le *DM*, aux formats supportés (texte, vidéo, son, image, *etc*.) et à la machine virtuelle de Java (kVM). Parmi les caractéristiques matérielles, nous pouvons mentionner l'écran, la plate-forme, *etc*. Le type de connexion, la qualité du service et le type d'interface sont des concepts liés aux caractéristiques (interface) réseau. La Figure 6.13 montre cette ontologie :

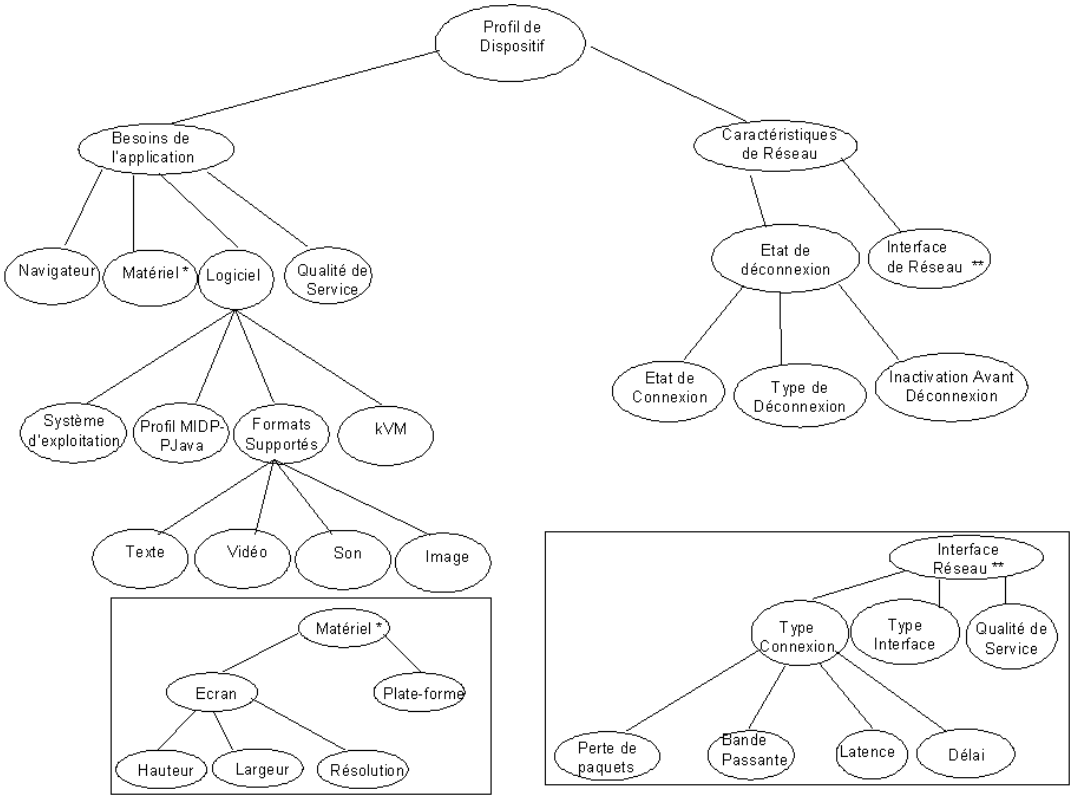


Figure 6.13. Ontologie du profil de dispositif.

[66] La description en *OWL* de ces quatre ontologies est donnée dans l'annexe 2. http://www-lsr.imag.fr/users/Angela.Carrillo/

La deuxième ontologie regroupe les caractéristiques de la session en cours (voir Figure 6.14) : l'utilisateur connecté, le type de dispositif d'accès, le moment auquel l'utilisateur s'est connecté et un ensemble d'activités. Ces activités sont composées de fonctionnalités (cf. section 8.1). Chaque activité se voit associer son état (abouti, échoué, suspendu, *etc*.).

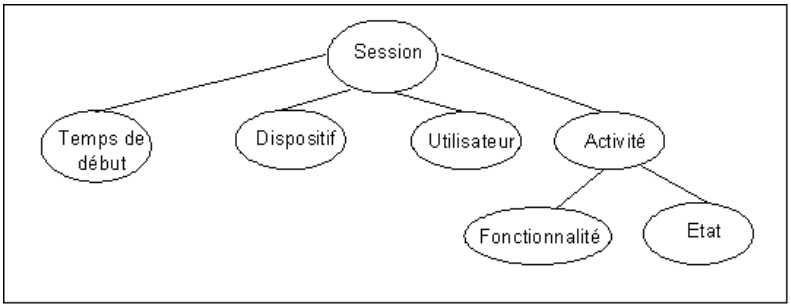


Figure 6.14. Ontologie de la session.

La troisième ontologie correspond à la spécification des caractéristiques de la localisation (voir Figure 6.15). Cette ontologie permet de spécifier les caractéristiques des localisations géodésique, logique et physique. Elle contient également des concepts liés à l'orientation de l'utilisateur et ses modifications. Les concepts liés à la *localisation géodésique* sont l'altitude, la latitude et la longitude ; ceux concernant la *localisation logique* sont le type de connexion du dispositif d'accès (par exemple, connexion filaire, connexion sans fil, adresse IP, *etc*.) ; et les concepts liés à la *localisation physique* sont : ville, pays, rue, *etc*. Parmi les concepts liés à l'*orientation* de l'utilisateur, l'ontologie permet de représenter sa vitesse de déplacement et ses changements d'orientation (direction, accélération).

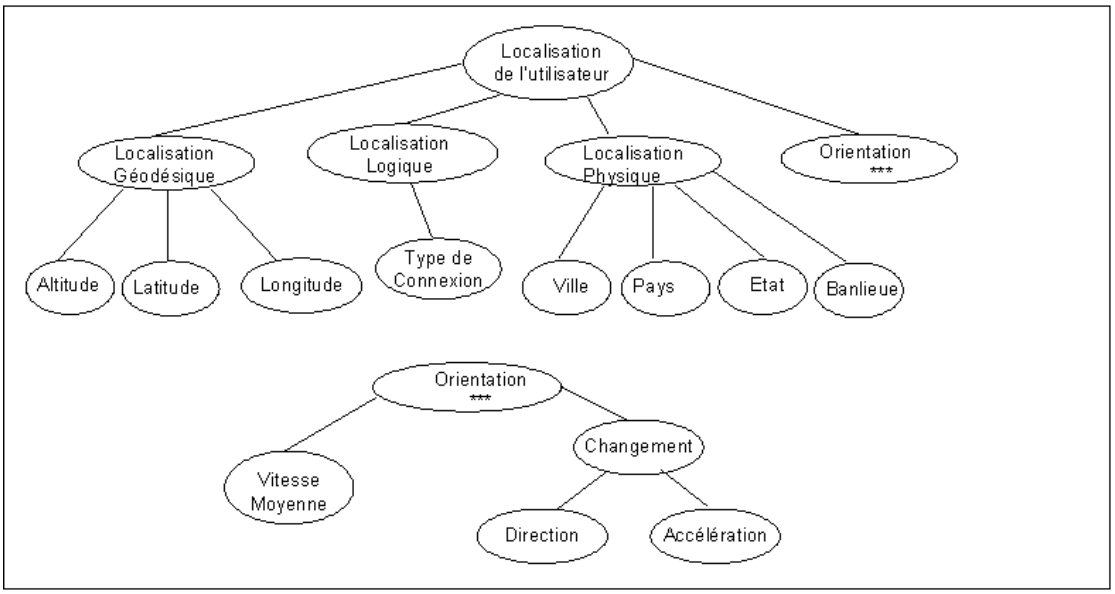


Figure 6.15. Ontologie de la localisation.

Les concepts concernant la localisation *géodésique*, *logique* et *physique*, présentés dans l'ontologie ci-dessus, peuvent être représentés en utilisant *GML*[67], le standard crée par l'*OpenGIS*[68] qui permet de définir une localisation de la manière suivante (cf. Figure 6.16) :

```
<element name="location" type="gml :LocationPropertyType"/>
<complexType name="LocationPropertyType">
    <sequence>
            <choice>
                    <element ref="gml:_Geometry"/>
                    <element ref="gml:LocationKeyWord"/>
                    <element ref="gml:LocationString"/>
                    <element ref="gml:Null"/>
            </choice>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

Figure 6.16. Représentation de la localisation en GML.

La valeur de la localisation peut être de type « *Geometry* », « *Location String* », « *Location Keyword* » ou « *Null* » (cf. Figure 6.17).

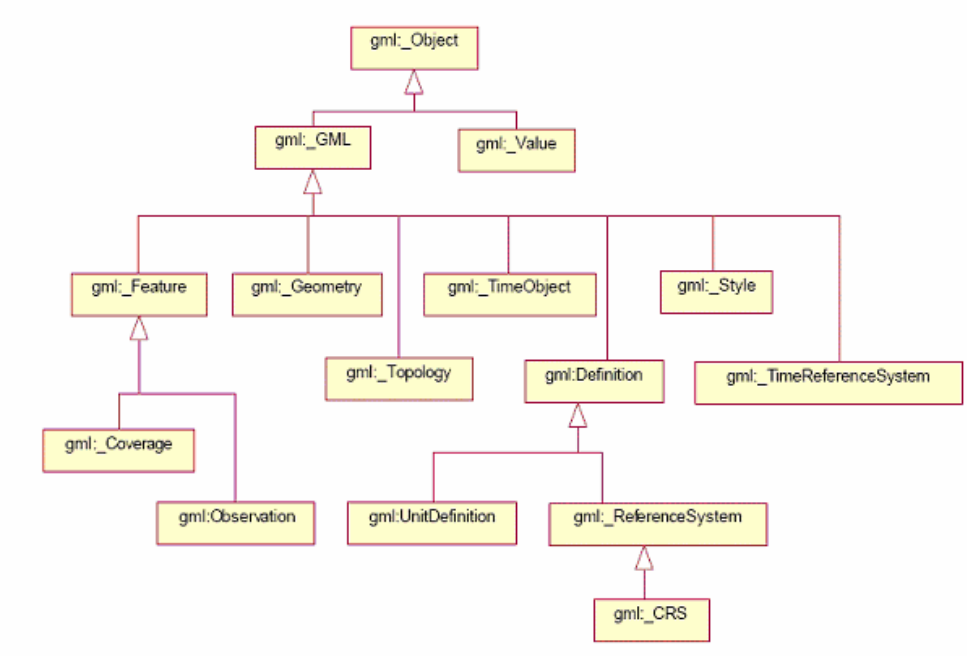


Figure 6.17. Hiérarchie de classes de GML.

[67] *GML* : Geography Markup Language : http://www.opengeospatial.org/standards/gml
[68] http://www.opengeospatial.org/

D'après l'*OpenGIS*, une « *Location String* » est un texte qui décrit la localisation de la manière suivante :

<element name="LocationString" type="gml :StringOrRefType"/>

Un « *Location Keyword* » est un code sélectionné d'une liste prédéfinie. Il est déclaré comme suit :

<element name="LocationKeyWord" type="gml :CodType"/>

Les exemples suivants (d'après l'*OpenGIS*) montrent la manière dont les trois types de localisation peuvent être représentés :

– La *localisation géodésique* peut être représentée à manière d'une gml « *Geometry* » en spécifiant un système de référence spatiale :

```
<gml :location>
    <gml :Point gml :id="point96" srsName="urn :EPSG :geographicCRS :62836405">
            <gml:pos>-31.936 115.834</gml:pos>
    </gml :Point >
< /gml :location >
```

– La *localisation physique* peut être représentée comme un « *Location KeyWord* » en utilisant un nom d'une source contrôlée :

```
<gml :location>
    <gml :LocationKeyWord        codeSpace=http://www.icsm.gov.au/icsm/cgna/index.html>
            Leederville
    </gml :LocationKeyWord>
</gml :location>
```

– La *localisation logique* peut être représentée comme une chaîne de caractères :

```
<gml :location>
        <gml :LocationString>Laboratoire LSR-IMAG </gml :LocationString>
</gml :location>
```

La quatrième ontologie décrit le profil de l'utilisateur (cf. Figure 6.18) qui est composé d'un ensemble de préférences classées en trois types (cf. section 8.1) : *activité*, *résultat* et *affichage*. Ces préférences représentent respectivement les activités qu'un utilisateur souhaite accomplir dans le système (activités composées de fonctionnalités exécutées d'une manière séquentielle, concurrente ou conditionnelle), les résultats attendus de ces activités, et la manière dont ces résultats sont affichés sur le dispositif d'accès.

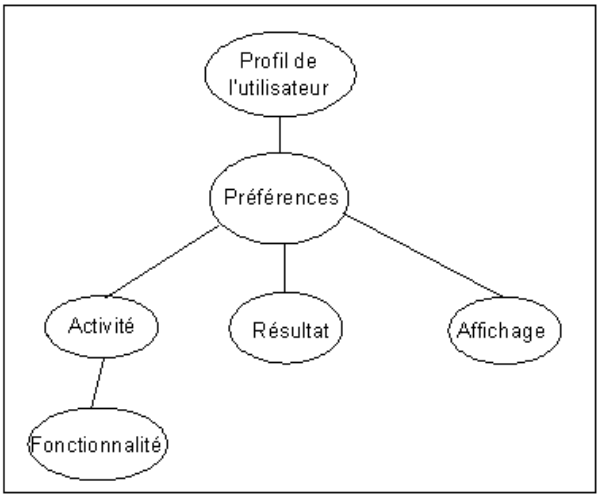


Figure 6.18. Ontologie de l'utilisateur.

6.2.4.2 Génération et gestion de bases de connaissances

Dans cette section, nous décrivons la gestion de la *connaissance* des agents du *SMA d'information* et du *SMA d'adaptation* de *PUMAS*. Cette connaissance, représentée à travers les ontologies présentées ci-dessus, est stockée dans des *bases de connaissances* sous la forme de pièces de connaissances nommées « *faits* ». Ces faits sont déclarés à la manière d'instances de *Templates JESS*[69] afin de représenter les préférences d'information de l'utilisateur, les caractéristiques du *DM*, les caractéristiques d'une session, *etc*. Nous avons choisi *JESS* car la description des faits détaille et étend les concepts définis dans les ontologies et son langage de règles fournit des mécanismes d'inférence de connaissances. Nous pouvons trouver des bibliothèques qui permettent d'intégrer des applications construites en Java (par exemple, des applications construites en utilisant *JADE*) avec *JESS*, et des outils qui facilitent (à travers des *plug-ins*, par exemple *Protégé*) l'exécution et la validation de règles écrites en *JESS*. *OWL* ne permet pas l'expression de règles ni ne fournit un mécanisme d'inférence de connaissances.

Nous commençons cette section avec une brève description de la notation de *JESS* pour la définition de faits.

a – Notation de JESS

Dans cette section, nous montrons la manière dont un Template *JESS* et leurs instances sont définis.

[69] *JESS* est un moteur de règles et un environnement de « *scripts* » utilisé pour la construction d'applications *JAVA* qui possède la capacité de « *raisonner* » en utilisant la connaissance fournie sous la forme de règles déclaratives. *http://herzberg.ca.sandia.gov/jess/*.

Afin de définir un fait en *JESS*, il est nécessaire de définir un *Template*. Un fait obtient son nom et la liste de « *slots*[70] » de son template.

JESS fournit la fonction « *deftemplate* » pour la définition d'un template (cf. Figure 6.19) :

```
(deftemplate template-name
    ["Documentation comment"]
    [(declare  (slot-specific TRUE | FALSE)
               (backchain-reactive TRUE | FALSE)
               (from-class class name)
               (include-variables TRUE | FALSE)
               (ordered TRUE | FALSE))]
    [extends template-name]
     (slot | multislot slot-name
         [(type ANY | INTEGER | FLOAT |
                NUMBER | SYMBOL | STRING |
                LEXEME | OBJECT | LONG)]
         [(default default value)]
         [(default-dynamic expression)])*)
```

Figure 6.19. Définition d'un Template JESS.

La déclaration d'un template inclut un nom, une documentation (optionnelle), une clause « *extends* » (optionnelle) et une liste de zéro ou plusieurs descriptions de slots, *etc*. Chaque slot peut inclure un type de qualificateur (type de donnée, par exemple, integer, float, string, *etc*.) ou une valeur par défaut. Dans la Figure 6.19, les valeurs par défaut sont écrites en gras.

Un exemple de template qui définit une session d'un utilisateur (appartenant à l'ontologie de la session, voir section précédente) est composé du nom de l'utilisateur (IDUtilisateur), du type de dispositif d'accès utilisé pour la connexion, de l'état de la connexion (par exemple, « connecté », « déconnecté », « suspendu », *etc*.), du temps de début et de l'ensemble des activités exécutées par l'utilisateur. Ce template est exprimé en *JESS* de la manière suivante :

(*deftemplate connexion*
(*slot IDUtilisateur*)
(*slot typeDM*)
(*slot état*)
(*slot tempsdébut*)
(*multislot activités*))

Afin de créer une instance d'un template, *JESS* utilise la fonction « *assert* » qui assigne à chaque slot une valeur. Afin de créer une instance du template décrit ci-dessus nous utilisons :

(*assert* (*connexion*
(*IDUtilisateur " Angela"*)
(*typeDM "Pocket PC 5500"*)
(*état "connecté"*)
(*slot tempsdébut "8 :00"*)
(*activités "consulter messages" "fixer rendez-vous"*))

[70] Un slot en *JESS* correspond à une caractéristique (ou attribut) d'un fait.

Puisqu'il est coûteux d'écrire une fonction « *assert* » pour chaque instance d'un template, *JESS* fournit la fonction « *deffacts* » qui construit une liste de faits. Cette liste possède un nom. Par exemple, pour créer plusieurs instances du *template* connexion, nous pouvons utiliser la fonction « *deffacts* » de la manière suivante :

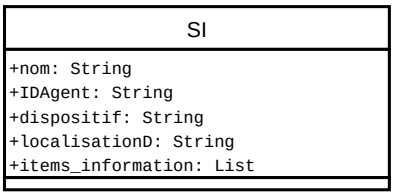
(*deffacts plusieurs_connexions*
(*connexion* (*IDUtilisateur " Angela "*) (*typeDM "Pocket PC 5500"*) (*état "connecté"*) (*tempsdébut "8 :00"*) (*activités "consulter messages" "fixer rendez-vous"*))
(*connexion* (*IDUtilisateur " Fernando"*) (*typeDM "Nokia 2000"*) (*état "déconnecté"*) (*tempsdébut "5 :00"*) (*activités "consulter messages"*)))

Dans les sections suivantes, nous utilisons ces trois fonctions de *JESS* (*deftemplate*, *assert* et *deffacts*) afin de décrire la connaissance du *SMA d'information* et celle du *SMA d'adaptation*.

b – Connaissances du SMA d'information

Les *agents de routage* stockent dans leurs bases de connaissances un fait pour chaque *Système d'Information* (*SI*). Un *agent de routage* exploite ces faits afin de rediriger les requêtes de l'utilisateur. Un fait représentant un *SI* décrit ses caractéristiques : son nom, l'information qu'il gère, le type de dispositif sur lequel il est exécuté (par exemple, un serveur, un *DM*), la localisation du dispositif (*localisationD*) et l'agent (*agent de SI*) associé à ce *SI* auquel doivent être adressées les requêtes. Le fait suivant définit un *SI* et est représenté par un template *JESS*[71] :

(*deftemplate SI*
(*slot nom*)
(*slot IDAgent*)
(*slot dispositif*)
(slot *localisationD*)
(*multislot items_information*))



Le *template* concernant le *SI* peut être utilisé pour définir le *SI* de la pharmacie d'un hôpital. Le *SI* est nommé « *SIPharmacie* » et il s'exécute sur un serveur. L'« *ASIPharmacie* » est l'*agent de SI* qui s'exécute sur ce *SI*. Le *SI* de la pharmacie contient l'information sur les médicaments prescrits à un patient :

(*assert* (*SI* (*nom SIPharmacie*)
(*agentID ASIPharmacie*)
(*dispositif serveur*)
(*localisationD " 45°11'16"N, 5°43'37"E "*)
(*items_information "médicaments prescrits"*)))

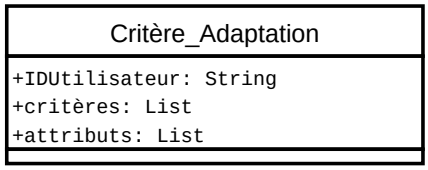
La localisation du *SI* pourrait changer, notamment si ce *SI* s'exécute sur un *DM*. L'*agent de routage* peut être informé des changements de localisation du *SI* à travers l'*agent de SI* qui s'exécute sur ce *SI*. Un autre aspect à notifier à l'*agent de SI* est la modification (par exemple, l'ajout ou l'élimination d'un item) de la liste d'items d'information.

[71] Afin de clarifier la définition d'un concept comme un *template JESS*, nous incluons également sa définition comme une classe *UML*.

c – Connaissances du SMA d'adaptation

Les requêtes dépendent d'un ou plusieurs critères dans le processus d'adaptation : localisation de l'utilisateur, historique dans le système, activités développées pendant une période de temps, orientation de déplacement, préférences de confidentialité, *etc*. Un *Critère_Adaptation* est défini de la manière suivante :

(*deftemplate Critère_Adaptation*
(*slot IDUtilisateur*)
(*multislot critères*)
(*multislot attributs*))

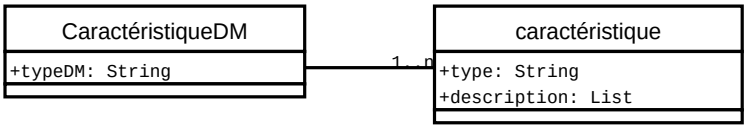


Un exemple d'un *Critère_Adaptation* exprimant que toutes les requêtes du *Docteur Thierry Dupont* dépendent de sa localisation, spécialement lorsqu'il est à l'*Hôpital Nord* est donné par :

(*assert* (*Critère_Adaptation*
(*IDUtilisateur "Docteur Thierry Dupont"*)
(*critères localisation*)
(*attributs "Hôpital Nord"*)))

L'*agent de filtre d'affichage* gère une *base de connaissances* contenant l'information générale sur les caractéristiques des différents types de *DM* (par exemple, les formats de fichiers supportés). Chaque *CaractéristiqueDM* est définie à travers un fait et est représentée de la manière suivante :

(*deftemplate*
CaractéristiqueDM
(*slot typeDM*)
(*multislot caractéristique*))



où chaque caractéristique est représentée aussi comme un fait de la manière suivante :

(*deftemplate caractéristique*
(*slot type*)
(*multislot description*))

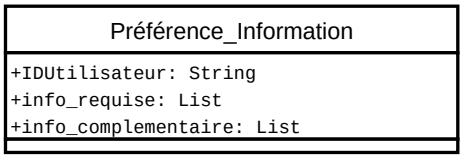
L'exemple ci-dessous décrit un fait pour une *CaractéristiqueDM* correspondant aux formats des fichiers supportés par un *Pocket PC hp IPAQ h5550* selon le type de réseau. Nous supposons que l'affichage vidéo n'est pas supporté à travers un *réseau Wi-Fi*, ni celui de plusieurs images en utilisant un réseau *Bluetooth* :

(*deffacts CaractéristiqueDM*
(*MDType "PocketPC hpIPAQ h5550")*
(*caractéristique* (*type "video non supporté"*) (*description "sur des réseaux Wi-Fi"*))
(*caractéristique* (*type "plusieurs images"*) (*description "sur Bluetooth"*)))

Pour sa part, l'*agent de filtre de contenu* gère une *base de connaissances* contenant les préférences des utilisateurs. Dans la section 8.1, nous présentons en détail les préférences de l'utilisateur vis-à-vis des activités qu'il souhaite accomplir dans le système, les résultats attendus de ces activités et la manière dont les résultats sont affichés. Ici, nous nous limitons aux préférences d'information de l'utilisateur. Une *Préférence_Information* exprime le besoin

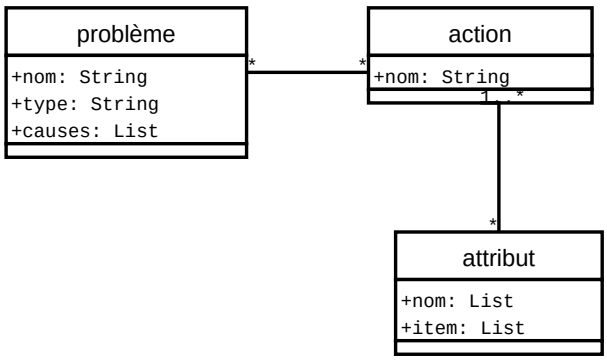
d'information de l'utilisateur à deux niveaux : prioritaire (info_requise) pour l'utilisateur, et additionnelle (info_complementaire) qu'il est souhaitable d'obtenir chaque fois qu'une information « *prioritaire* » est demandée. Le fait correspondant à une *Préférence_Information* composé d'un *IDUtilisateur* (identifiant le propriétaire de cette préférence), d'une information requise et d'une information complémentaire, est défini de la manière suivante :

(*deftemplate Préférence_Information*
(*slot IDUtilisateur*)
(*multislot info_requise*)
(*multislot info_complementaire*))



Afin de décrire la manière dont l'utilisateur souhaite obtenir les résultats à partir d'une de ses requêtes, nous définissons des *actions*. Elles sont aussi utilisées pour décrire la manière dont le système doit répondre dans le cas où un problème apparaît. Nous définissons un *problème* comme un événement non souhaitable durant l'exécution d'une action, ou qui est la cause d'un échec (par exemple, si le *DM* ne peut pas afficher une image). Chaque problème est défini comme un fait et est représenté de la manière suivante :

(*deftemplate problème*
(*slot nom*)
(*slot type*)
(*multislot causes*))



où *nom* correspond à une description du problème, *type* peut être choisi parmi une liste définie par les concepteurs de l'application (par exemple, *incompatibilité, SI non disponible, agent non disponible*), et les *causes* correspondent à une liste de causes potentielles de ce problème (par exemple, le *DM* ne prend pas en charge un format de fichier spécifique). Un fait, qui définit le *problème* lié à une localisation spécifique de l'utilisateur, hors de portée d'un réseau sans fil, est décrit par :

(*assert* (*problème*
(*nom "connexion hors portée"*)
(*type "défaut d'accès"*)
(*causes "utilisateur situé hors portée" " réseau hors service"*)))

La manière dont le système doit répondre en présence de problèmes est représentée à travers d'actions. Chaque *action* est définie comme un fait et est représentée de la manière suivante :

(*deftemplate action*
(*slot nom*)
(*multislot attribut*))

Dans cette définition, *nom* réfère à une action choisie parmi une liste définie (par exemple, « *afficher* », « *sauvegarder* », « *transférer fichier* », « *annuler* ») et chaque action possède une liste d'attributs. Par exemple, le fait représentant l'action « *afficher* » a comme propriétés l'*ordre* (dans lequel afficher l'information), le *format* (auquel l'afficher) et le *type de fichier* (à utiliser pour l'afficher), est :

(*assert* (*action*
(*nom montrer*)
(*attributs "ordre" " format" "type fichier"*)))

Un *attribut* complexe peut être défini comme un fait :

(*deftemplate attribut*
(*slot nom*)
(*multislot liste*))

Un exemple d'attribut définissant l'ordre dans lequel l'information doit être affichée, est :

(*assert* (*attribut*
(*nom ordre*)
(*items "analyses médicales" "régime" "médicaments prescrits"*)))

L'ensemble de règles et de requêtes exprimées en utilisant *JESS* dans le processus d'adaptation de l'information est donné dans l'annexe 2[72].

6.3. Conclusion

Lors d'un processus de *recherche d'information* dans des environnements nomades, un utilisateur peut être confronté à plusieurs problèmes tels que : des contraintes d'accès aux *Systèmes d'Information* liées aux caractéristiques de réseaux et de son dispositif d'accès ; une quantité d'information qui n'est pas toujours pertinente par rapport à ses besoins d'information et à ses préférences, ni ne peut être affichée sur son dispositif d'accès mobile ; le manque de mécanismes pour la recherche d'information distribuée sur plusieurs systèmes d'information s'exécutant sur divers types de dispositifs (serveurs ou *DM*). Afin de résoudre ces problèmes, nous avons proposé *PUMAS*, un *framework* basé sur des agents qui fournit à l'utilisateur nomade l'information adaptée à ses caractéristiques et à celles de son dispositif d'accès. *PUMAS* offre aussi les moyens d'accéder à plusieurs *Systèmes d'Information* (exécutés sur des serveurs ou des *DM*) à travers différents types de dispositifs d'accès.

L'architecture de *PUMAS* est composée de quatre *SMA* :

- *Le SMA de connexion* qui fournit des mécanismes pour faciliter la connexion de différents types de *DM* aux *Systèmes d'Information.*
- *Le SMA de communication* qui assure une communication transparente entre le *DM* et le système, et applique un *filtre d'affichage* (à l'aide des agents du *SMA d'adaptation*)

[72] http://www-lsr.imag.fr/users/Angela.Carrillo/

afin de présenter l'information d'une manière adaptée en prenant en compte les contraintes techniques du *DM* de l'utilisateur.

- *Le SMA d'information* qui reçoit les requêtes des utilisateurs, les redirige aux *Systèmes d'Information* (*SI*) susceptibles d'apporter une réponse (par exemple, le *SI* le plus proche, le plus consulté), applique un *filtre de contenu* (à l'aide des agents du *SMA d'adaptation*) en tenant compte du profil de l'utilisateur dans le système et retourne les résultats au *SMA de communication.*
- *Le SMA d'adaptation* qui communique avec des agents des trois autres *SMA* afin d'échanger de l'information sur l'utilisateur, les caractéristiques de connexion et de communication, les caractéristiques du *DM*, *etc*.

La mobilité inhérente des utilisateurs nomades est supportée par des *agents ubiquitaires* : les *agents de DM* s'exécutant sur le *DM* de l'utilisateur et les *agents de SI* s'exécutant sur le même dispositif que le *SI* auquel ils appartiennent. Les agents ubiquitaires récupèrent l'information nécessaire et peuvent communiquer avec d'autres agents afin d'accomplir les tâches qui leur sont assignées.

PUMAS repose sur une architecture hybride *P2P*. Le noyau de *PUMAS* centralise les requêtes : d'une part, il est en charge du processus visant à obtenir l'information la plus pertinente (qui satisfait les besoins d'information de l'utilisateur), et d'autre part, il est en charge d'appliquer les filtres de *contenu* et d'*affichage* afin d'adapter les résultats.

Les caractéristiques principales des systèmes *P2P* reposant sur *PUMAS* sont : *i)* Un *DM* peut communiquer avec un *SI* spécifique (situé sur un serveur ou sur un *DM*) en passant l'identifiant de ce système comme un paramètre de la requête. L'*agent de routage* transmet alors la requête à ce *SI* spécifique (communication d'agent à agent), et *ii)* les agents ont l'autonomie de se connecter à *PUMAS* et de se déconnecter.

PUMAS gère quatre ontologies qui décrivent respectivement les caractéristiques liées à la session de l'utilisateur, son dispositif d'accès, ses préférences, et sa localisation. Les concepts appartenant à chacune de ces ontologies sont inspirés des travaux d'Indulska *et al.* [Indu03a]. Nous avons détaillé la définition de chacun de ces concepts en utilisant des pièces de connaissances (« *faits* ») stockées dans des bases de connaissances gérées par les agents de *PUMAS*. Dans ce chapitre, nous nous sommes focalisés sur la définition des faits qui concernent des buts d'adaptation de l'information pour les *SMA d'information* et *d'adaptation*.

Lorsqu'un utilisateur formule des requêtes, les résultats peuvent être détenus par différents *SI*. Le problème est alors de sélectionner les *SI* capables de répondre à ces requêtes et de compiler par la suite leurs résultats. Un processus de *routage de requêtes* adapté au contexte de *PUMAS* est présenté dans le chapitre suivant.

7. PROCESSUS DE ROUTAGE DE REQUETES DANS PUMAS

Nous présentons dans ce chapitre le processus de *routage de requêtes* dans *PUMAS*. Le processus se base sur l'analyse des requêtes et leur redirection vers les *Systèmes d'Information* (*SI*) capables d'y répondre. La sélection des *SI* repose sur leur aptitude à répondre à des critères d'adaptation tels que les préférences de l'utilisateur, ses activités, la localisation (celle de l'utilisateur, celle de ses collègues), *etc*. Nous décrivons par la suite l'impact des changements de localisation pour les requêtes sensibles à cet aspect. Nous terminons ce chapitre par la présentation de trois scénarios qui décrivent l'utilisation de *PUMAS* pour la connexion d'un utilisateur, l'envoi d'une requête et la réception de ses résultats.

7.1. Processus de routage de requêtes

Afin que dans *PUMAS* les résultats de requêtes puissent provenir de différents *SI*, il est nécessaire de disposer d'un mécanisme qui, d'un côté, analyse les requêtes pour choisir le(s) *SI* capable(s) d'y répondre, et, de l'autre, compile les résultats de ces requêtes. Ce mécanisme est appelé le « *processus de routage de requêtes* ». Dans cette section, nous expliquons et illustrons chaque activité de ce processus. Le *routage de requêtes* dans *PUMAS* est accompli par les *agents de routage* (appartenant au *SMA d'information*) qui reçoivent les requêtes d'information auxquelles sont ajoutées les caractéristiques de l'utilisateur et celles de son *DM*. Pour ce processus, nous nous sommes inspirés des trois activités présentées dans le travail de Xu *et al.* [XuLi99] que nous avons adaptées au contexte de *PUMAS*. Ces activités sont décrites et illustrées en utilisant des scénarios construits sur les activités de consultation d'un médecin.

7.1.1 Analyse de la requête

Cette activité est liée à l'éventuelle décomposition d'une requête en sous-requêtes. L'*agent de routage* analyse la complexité de la requête. Une requête est considérée comme « *simple* » si elle peut être traitée par seulement un *agent de SI* et « *complexe* » si plusieurs *agents de SI* sont requis. Cette analyse est plus précisément basée sur les faits relatifs aux *SI*, stockés dans la *base de connaissances* de l'*agent de routage*. L'*agent de routage* analyse également les critères d'adaptation d'une requête (par exemple, la localisation, les activités de l'utilisateur), la liste des destinataires d'une requête, *etc*. Après cette analyse, l'*agent de routage* décide s'il doit ou non décomposer la requête en sous-requêtes.

Supposons que le *docteur Dupont* souhaite consulter *les résultats des analyses médicales d'un patient, son régime alimentaire et les médicaments qui lui ont été prescrits*. Supposons qu'aucun *SI* ne puisse répondre entièrement à cette requête. Cette requête est considérée comme une requête « *complexe* » qui est décomposée par l'*agent de routage* en trois sous-requêtes concernant respectivement les « *analyses médicales* », le « *régime* » et les « *médicaments prescrits* ». Une telle décomposition est basée sur la connaissance stockée dans la *base de connaissances* de l'*agent de routage* contenant l'information sur les *SI*. Supposons que l'*agent de routage* ait la connaissance des *SI* de la pharmacie, du nutritionniste, du laboratoire d'analyses médicales et des médecins de l'hôpital. Les *SI* sont décrits ci-dessous en *JESS* :

(*assert* (*SI*
(*nom SIPharmacie*)
(*agentID ASIPharmacie*)
(*dispositif serveur*)
(*localisationD " Hôpital Nord "*)
(*items_information "médicaments prescrits patients" "dosage" "médicaments"*)))

(*assert* (*SI*
(*nom SINutritionniste*)
(*agentID ASINutritionniste*)
(*dispositif DM*) (*localisationD " Hôpital Nord "*)
(*items_information "régimes patients" "régimes généraux" "rendez-vous patients"*)))

(*assert* (*SI*
(*nom SILaboratoireAnalysesMédicales*)
(*agentID ASILaboratoireAnalysesMédicales*)
(*dispositif serveur*)
(*localisationD " Hôpital Nord "*)
(*items_information "analyses médicales" "réactifs" "analyses médicales patients" "recommandations analyses médicales"*)))

(*assert* (*SI*
(*nom SIMédecins*)
(*agentID ASIMédecins*)
(*dispositif serveur*)
(*localisationD " Hôpital Nord "*)
(*items_information "médecins" "rendez-vous patients" "chirurgies" "emploi du temps"*)))

Afin de décomposer la requête, l'*agent de routage* identifie les items de la requête [$item_1$, $item_2$... $item_n$] lors de l'activité d'analyse de la requête. Dans notre exemple, la requête correspond à : « *les résultats des analyses médicales d'un patient, son régime alimentaire et les médicaments qui lui ont été prescrits* », et l'$item_1$ correspond à « *analyses médicales* », l'$item_2$ à

« *régime* » et l'item$_3$ à « *médicaments prescrits* ». Ensuite, cet agent cherche pour chaque item, le(s) *SI* qui gère(nt) un *item_information* équivalent[73].

Nous donnons ci-dessous, un algorithme de correspondance qui génère une liste *LSI* contenant des tuples (*SI*, <liste d'items de la requête gérés par le *SI*>). Tout d'abord, la liste *LSI* est vide (cf. ligne (1)). Ensuite, la variable *nSI* contentant le nombre de *SI* gérant les items équivalents à ceux de la requête (cf. ligne (2)) est initialisée. Pour chaque item de la requête (item$_i$ avec i ∈ [1, n], cf. lignes (3) à (18)), on cherche les SI_j (SI$_j$ avec j ∈ [1, s]) gérant des items d'information équivalents à celui en cours d'analyse. La méthode « *Comparer* » teste si les items sont équivalents (cf. ligne (9)). S'ils sont équivalents, l'algorithme augmente la variable *nSI*, et ajoute à *LSI* un tuple dont le premier terme correspond à SI_j et le deuxième correspond à l'item analysé (cf. lignes (10) et (11)). Lorsque tous les items de la requête ont été analysés, l'algorithme vérifie la valeur de *nSI* (cf. lignes (19) à (27)). Si cette valeur est nulle, cela signifie qu'il n'existe aucun *SI* qui gère des items équivalents à ceux de la requête. Dans le cas contraire, l'algorithme analyse la liste *LSI* afin de connaître le nombre de *SI* différents gérant des items équivalents à ceux de la requête afin d'attribuer le type de la requête (« *simple* » ou « *complexe* »). Ce nombre est calculé en utilisant la méthode « *compterDifferents* » (cf. ligne (22)). Finalement, l'algorithme utilise la méthode « *Compacter* » qui laisse dans *LSI* un seul tuple par *SI*. Le deuxième terme de ce tuple correspond à tous les items de la requête gérés par *SI*.

[73] Deux items sont « *équivalents* » s'ils sont égaux sémantiquement (c'est-à-dire, si les deux items ont la même signification) ou syntaxiquement (par exemple, égalité de chaînes de caractères). La relation d'équivalence est définie par les concepteurs des applications qui mettent en œuvre l'algorithme de correspondance expliqué dans cette section.

```
(1) Initialiser la liste de SI répondant particulièrement à la requête (LSI) // LSI est initialisé à la liste vide
(2) nSI← 0 ; // compteur des SI gérant des items équivalents à ceux de la requête
(3) i← 1 ; // index sur les items de la requête
(4) Tant que i ∈ [1, n] faire // index sur les items de la requête
(5)         j← 1 ; // index sur les SI
(6)         Tant que j ∈ [1, s] faire // index sur les SI connus par l'agent de routage
(7)                 m← 0 ; // index sur les items d'information du SIj
(8)                 Tant que m < taille (liste d'items d'information gérés par SIj) faire
(9)                         Si Comparer (itemi, item_informationm géré par SIj) alors
(10)                        nSI ← nSI +1 ;
(11)                        LSI← LSI ⊕ (SIj, <itemi>) ; // ajouter le tuple a LSI
(12)                        Fin Si
(13)                m ← m + 1 ;
(14)                Fin Tant que
(15)        j ← j + 1 ;
(16)        Fin Tant que
(17) i ← i + 1 ;
(18) Fin Tant que
(19) Si nSI est égal à 0 alors
(20)        type_requête ← "sans réponse"
(21) sinon
(22)        sid← compterDifférents (LSI) ; // compte le nombre de SI différents appartenant à LSI
(23)        Si sid est égal à 1 alors
(24)                type_requête ← "simple"
(25)        sinon  // sid est supérieur à 1
(26)                type_requête ← "complexe"
(27)        Fin Si
(28) Fin Si
(29) LSI ← Compacter (LSI)
```

Afin de clarifier le fonctionnement des fonctions « *compterDifférents* » et « *Compacter* », nous les illustrons à travers un exemple :

Supposons que l'*agent de routage* ait identifié les items : i_1, i_2, i_3, i_4 et i_5, et qu'il connaisse les *SI* suivants : SI_1, SI_2, SI_3, SI_4, SI_5 et SI_6. Après la comparaison des items (cf. lignes (4) à (19)), nous supposons que *LSI* soit composée des tuples suivants :

(SI_1, <i_1>)	(SI_3, <i_1>)	(SI_5, <i_1>)	(SI_1, <i_2>)	(SI_3, <i_2>)
(SI_4, <i_2>)	(SI_5, <i_2>)	(SI_1, <i_3>)	(SI_3, <i_3>)	(SI_4, <i_4>)
(SI_5, <i_4>)	(SI_1, <i_5>)	(SI_3, <i_5>)	(SI_4, <i_5>)	(SI_5, <i_5>)

Le résultat de la méthode « *compterDifférents* » est 4 car dans les tuples n'apparaissent que SI_1, SI_3, SI_4 et SI_5 (SI_2 et SI_6 n'apparaissent pas).

La méthode « Compacter » laisse un seul tuple par *SI*. Apres l'exécution de cette méthode, la *LSI* aura les tuples suivants :

(SI_1, <i_1, i_2, i_3, i_5>)
(SI_3, <i_1, i_2, i_3, i_5 >)
(SI_4, <i_2, i_4, i_5 >)
(SI_5, <i_1, i_2, i_4, i_5>)

Une optimisation de l'algorithme qui n'utilise pas la méthode « *Compacter* » est la suivante :

```
(1) Initialiser la liste de SI répondant particulièrement à la requête (LSI) // LSI est initialisée à la liste vide
(2) nSI← 0 ; // compteur des SI gérant des items équivalents à ceux de la requête
(3) j← 1 ; // index sur les SI
(4)  Tant que j ∈ [1, s] faire // index sur les SI connus par l'agent de routage
(5)          m← 0 ; // index sur les items d'information du SIj
(6)          Initialiser la liste d'items SIj (LItem) qui font la correspondance avec les items de la requête
                                                                          // LItem est initialisée à la liste vide
(7)          Tant que m < taille (liste d'items d'information gérés par SIj) faire
(8)                  i← 1 ; // index sur les items de la requête
(9)                  Tant que i ∈ [1, n] faire // index sur les items de la requête
(10)                         Si Comparer (itemi, item_informationm géré par SIj) alors
(11)                                 LItem← LItem ⊕ itemi ; // ajouter l'item à LItem
(12)                         Fin Si
(13)                         i ← i + 1 ;
(14)                 Fin Tant que
(15)                 m ← m + 1 ;
(16)         Fin Tant que
(17)         Si LItem n'est pas vide alors
(18)                 LSI← LSI ⊕ (SIj, LItem) ; // ajouter le tuple à LSI
(19)                 nSI ← nSI +1 ;
(20)         Fin Si
(21)         j ← j + 1 ;
(22) Fin Tant que
(23) Si nSI est égal à 0 alors
(24)         type_requête ← "sans réponse"
(25) sinon
(26)         sid← compterDifférents (LSI) ; // compte le nombre de SI différents appartenant à LSI
(27)         Si sid est égal à 1 alors
(28)                 type_requête ← "simple"
(29)         sinon  // sid est supérieur à 1
(30)                 type_requête ← "complexe"
(31)         Fin Si
(32) Fin Si
```

Dans l'exemple, la *LSI* est composée de :

(*SIPharmacie, <"médicaments prescrits">*)
(*SINutritionniste, <"régime">*)
(*SILaboratoireAnalysesMédicales, <"analyses médicales">*)

Après l'exécution de l'algorithme, nous pouvons donc conclure que la requête est « *complexe* ».

Un item de la requête peut être géré par plusieurs *SI*, il est donc nécessaire de sélectionner les plus appropriés pour leur répondre. La section suivante décrit cette sélection.

7.1.2 Sélection de Systèmes d'Information

Une requête peut être redirigée vers un agent spécifique ou vers un groupe d'agents. Si les destinataires d'une requête sont connus, la sélection est relativement aisée. Autrement, l'*agent de routage* sélectionne les *SI* et compose le *réseau de voisins* (en prenant en compte la *LSI*, produite durant l'activité précédente). Cette sélection reprend les idées de Yang *et al*. [Yang04]. Ces auteurs proposent une approche efficace de *routage de requêtes* pour la récupération d'information en réseaux *P2P* non structurés. Les politiques de *routage de requêtes* sont utilisées afin de déterminer à quels nœuds une requête devrait être envoyée dans le réseau. Ce

travail introduit le concept de *Guide de Routage* (« *Routing Guide* »). Celui-ci se base sur les résultats des requêtes précédemment traitées. Il est utilisé afin de déterminer les différents chemins que les requêtes peuvent suivre vers les *SI* correspondants. Pour le processus de récupération de l'information dans un système *P2P*, chaque nœud possède une collection de données partagée avec d'autres nœuds. Lorsqu'un utilisateur soumet une requête, son nœud devient la source de la requête (expéditeur) et peut envoyer des messages (incluant la requête) à plusieurs de ses voisins. Lorsqu'un voisin reçoit le message avec la requête, il la traite en utilisant tout d'abord son information locale. Si le nœud trouve quelques résultats, il les retourne au nœud expéditeur.

Dans notre proposition, un pair est dit *voisin* d'autres pairs, s'il satisfait un ensemble de caractéristiques (des critères définis dans les *préférences de l'utilisateur* d'une application), telles qu'une localisation proche, des activités semblables, un rôle similaire, une connaissance similaire, des collègues travaillant dans le même groupe, *etc*. Les caractéristiques ne sont cependant pas restreintes à des critères de proximité.

Dans la suite, nous présentons trois cas à la base de la constitution d'un réseau de voisins dans lequel chaque nœud est un S*ystème d'Information* :

Le *cas* le plus simple est celui où une requête peut être seulement traitée par un agent. L'*agent de routage* utilise sa *base de connaissances* afin de contacter le *SI* pouvant répondre à la requête.

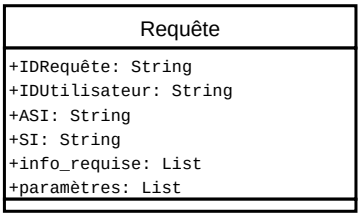
Dans le *deuxième cas*, nous considérons qu'un ou plusieurs agents peuvent répondre à la même requête. La manière la plus simple de composer ce réseau est de grouper tous ces agents. Ce rassemblement est utile lorsque l'*agent de routage* ne possède aucune information sur les *SI* ou lorsque c'est la première fois que l'*agent de routage* travaille avec les voisins. Afin d'éviter des communications inutiles, redondantes ou inutilisables, et de sélectionner les voisins les plus pertinents, l'*agent de routage* applique des critères d'adaptation de la requête. Par exemple, si le critère est la *localisation*, le réseau est composé des *voisins les plus proches*; si les requêtes de l'utilisateur dépendent de ses *requêtes précédentes*, l'*agent de routage* doit les rediriger aux voisins les plus dignes de confiance (« *trusted* ») ; si le critère est la *similarité*, le réseau doit être composé des voisins avec un *profil similaire*, *des tâches similaires*, *etc*. S'il n'y a pas de critères établis, l'*agent de routage* analyse le niveau de confiance (« *trust* ») de ses voisins. L'*agent de routage* associe un niveau de confiance à chaque voisin à partir des réponses précédentes comme proposé par Agostini *et al.* [Agos04]. Dans le travail de ces auteurs, lorsqu'un pair propage une requête, il analyse les résultats reçus et accroît la confiance des pairs qui répondent avec les contenus sémantiques les plus appropriés (le processus d'envoi de requêtes d'un pair vers d'autres est décrit par les auteurs). Après la formulation de la requête, un pair nommé « *chercheur* » (« *seeker* ») vérifie quels pairs sont connectés (« *actifs* ») et choisit, parmi ceux-ci, les pairs auxquels envoyer la requête. Une stratégie de *confiance* et de *réputation* (« *Trust and Reputation* », cf. 2.2.3) est utilisée par les « *chercheurs* » afin de cataloguer (évaluer) les fournisseurs potentiels et de choisir celui (ceux) qui répondra(ont) le mieux à la requête.

La stratégie de *confiance*[74] et de *réputation*[75] proposée par Agostini *et al.* [Agos04] repose sur le processus suivant : le *chercheur* envisage le problème de sélection de pairs capables de répondre à la requête *Q* avec la plus haute probabilité d'y bien répondre, ou quels sont les pairs les plus dignes de confiance pour répondre à la requête. Afin de décider, le *chercheur* construit et gère une liste <*p1, p2,…pk*> de pairs dignes de confiance à qui soumettre la requête. La liste est ordonnée en utilisant un niveau de confiance décroissant. Le *chercheur* interroge les pairs à tour de rôle, en commençant par *p1* et jusqu'à ce qu'il reçoive des réponses pertinentes. Cette liste de pairs dignes de *confiance* peut évoluer. La *confiance* d'un pair est acquise par sa *réputation*.

Dans le *troisième cas*, une requête a été décomposée en plusieurs sous-requêtes lors de l'étape d'analyse. L'*agent de routage* analyse quels agents peuvent répondre à chaque sous-requête. Ces agents constituent le *réseau de voisins*. Pour chaque sous-requête, on effectue la sélection des *SI* (soit avec le premier ou le deuxième cas). Finalement, le *réseau de voisins* est composé de l'agrégation des différents sous-réseaux générés pour chaque sous-requête.

Dans l'exemple de la requête du *docteur Dupont*, l'*agent de routage* prend en compte la *LSI* provenant de l'activité précédente (l'analyse de la requête). L'*agent de routage* sélectionne comme *SI* celui de la *Pharmacie*, celui du *Laboratoire d'Analyses Médicales* et celui des *Nutritionnistes* de l'hôpital car il n'existe qu'un *SI* capable de répondre à chaque sous-requête. Dans le cas où plusieurs *SI* auraient pu répondre à une sous-requête, l'*agent de routage* aurait du utiliser une stratégie de *confiance et réputation* comme celle expliquée ci-dessus. Pour cet exemple, l'*agent de routage* génère aussi les trois sous-requêtes qu'il redirige aux *agents de SI* de ces *SI* :

(*assert* (*Requête*
(*IDRequête Requête1*)
(*IDUtilisateur "Docteur Thierry Dupont"*)
(*ASI "ASILaboratoireAnalysesMédicales"*)
(*SI "SILaboratoireAnalysesMédicales"*)
(*info_requise "Analyses Médicales"*)
(*paramètres "nompatient" "date"*)))



(*assert* (*Requête*
(*IDRequête Requête2*)
(IDUtilisateur "Docteur Thierry Dupont")
(*ASI "ASINutritionniste"*)
(*SI "SINutritionniste"*)
(*info_requise "régime"*)
(*paramètres "nompatient" "date"*)))

[74] La *confiance* portée à un agent est le degré de sécurité estimé par rapport à la qualité de ses réponses. La confiance est basée sur sa *réputation*.

[75] La *réputation* d'un agent est l'opinion que les autres agents ont sur lui par rapport aux résultats fournis des requêtes en considérant des critères tels que la vitesse de réponse, la qualité des données, l'exhaustivité des réponses, la pertinence des réponses (celle-ci est évaluée directement par l'utilisateur ou assignée en fonction d'une votation, une pondération, une fonction de calcul, *etc*.), entre autres. La réputation est représentée comme une valeur (dans une échelle qualitative ou quantitative) calculée par chaque agent en utilisant une fonction.

(*assert* (*Requête*
(*IDRequête Requête3*)
(*IDUtilisateur "Docteur Thierry Dupont"*)
(*ASI "ASIPharmacie"*)
(*SI "SIPharmacie"*)
(*info_requise "médicaments prescrits"*)
(*paramètres "nompatient" "date"*)))

L'*agent de routage* doit également analyser le niveau de confiance associé à ces voisins. Si c'est la première fois que l'*agent de routage* exécute cette requête ou qu'il travaille avec ces *SI* ou encore, que les niveaux de confiance des voisins sont égaux, l'*agent de routage* leur envoie la requête à travers un message transmis en « *broadcast* » car cet agent ne dispose pas de critères de classement de ses voisins par rapport à leur réputation.

7.1.3 Redirection de la requête

Une fois que l'*agent de routage* a identifié les *SI* (*voisins*) potentiels, il redirige la requête à ses *voisins* en leur envoyant un message. L'*agent de routage* peut utiliser un message orienté (pour les récepteurs spécifiques) ou un « *broadcast* » à tous les voisins. L'*agent de routage* envoie le message d'une manière séquentielle, en commençant par l'agent le plus digne de confiance. La réponse à la requête sera celle du premier agent qui répond. Si l'*agent de routage* ne reçoit aucune réponse, l'utilisateur sera informé de l'échec de la requête.

Si l'*agent de routage* connaît le voisin pour chaque sous requête ($requête_1$, …, $requête_N$), il envoie le message orienté au voisin approprié. Par exemple, pour la requête du *docteur Dupont* à propos du dossier du patient (les analyses médicales, le régime et les médicaments prescrits), l'*agent de routage* envoie la *Requête1* à l'*ASILaboratoireAnalysesMédicales,* la *Requête2* à l'*ASINutritionniste* et la *Requête3* à l'*ASIPharmacie* (cf. section précédente).

L'*agent de routage* doit ensuite compiler les réponses obtenues des différents agents et sélectionner les plus pertinentes en tenant compte des critères d'adaptation établis. Tout d'abord, l'*agent de routage* sélectionne les réponses provenant des voisins les plus dignes de confiance, puis, il vérifie si les réponses sont conformes aux critères d'adaptation. Finalement, l'*agent de routage* envoie les résultats à l'*agent relais* qui démarre le processus de filtrage de contenu qui s'appuie sur le profil de l'utilisateur (cf. section 8.6).

7.2. Impact des changements de localisation

La composition du *réseau de voisins* (activité 2 du processus de *routage de requêtes*, cf. section 7.1.2) peut être hautement dynamique puisque les requêtes peuvent être dépendantes de la localisation de l'utilisateur qui formule la requête et de la localisation des *SI* (surtout si elles sont des *DM*). Dans le cadre du routage, nous considérons trois cas de composition du réseau de voisins :

1. Prendre uniquement en compte les changements concernant la localisation de la source de la requête, en supposant les récepteurs fixes.
2. Tenir uniquement compte des changements de localisation des récepteurs de la requête, en supposant la source de la requête fixe.

3. Tenir compte des changements de localisation de la source et des récepteurs de la requête.

L'*agent de routage* peut connaître la localisation de l'agent source de la requête en interrogeant l'*agent coordinateur* (appartenant au *SMA de communication*) qui connaît tous les agents et leurs services. La localisation des agents récepteurs de la requête peut être fournie à l'*agent de routage* par les *agents de SI*.

Dans le premier cas évoqué ci-dessus, il est nécessaire de connaître son orientation, ainsi que sa vitesse de déplacement. L'*agent proxy* (appartenant au *SMA de communication*) réclame les caractéristiques de la localisation à l'*agent contrôleur de connexion*s (appartenant au *SMA de connexion*). Ce dernier agent obtient la localisation de la source auprès de l'*agent de DM* (qui s'exécute sur le *DM* de l'utilisateur) vérifiant les changements de cette localisation. Avec la nouvelle localisation, l'*agent contrôleur de connexions* envoie un message à l'*agent de DM* pour savoir si les résultats qui vont être affichés sont encore valables. Si c'est le cas, le réseau de voisins reste le même. Autrement, l'*agent de routage* doit composer un autre réseau de voisins à partir de la nouvelle localisation de la source de la requête.

Dans le deuxième cas, le réseau de voisins peut être construit dans la mesure où l'*agent de routage* a demandé aux *agents de SI* leur localisation. Dans ce cas, il est nécessaire de tenir compte de l'orientation et de la vitesse de déplacement des agents récepteurs.

Dans le dernier cas, les solutions établies pour les deux premiers cas sont considérées.

7.3. Scénarios d'utilisation de PUMAS

Dans cette section, nous présentons les scénarios associés à la soumission d'une requête à *PUMAS* : la première connexion à *PUMAS*, l'envoi de la requête de l'utilisateur à *PUMAS*, et l'envoi des résultats depuis *PUMAS* vers l'utilisateur. Dans notre proposition, les interactions entre les agents reposent sur les messages échangés en tenant compte des *actes de communication* présentés par Odell *et al.* [Odel01] (par exemple, « *confirm* », « *inform* », « *propose* », « *query* », « *subscribe* », « *propagate* », *etc*., voir section 3.3.2).

7.3.1 Scénario associé à la connexion

Lorsqu'un utilisateur se connecte la première fois au système en utilisant son *DM*, l'*agent de DM*, s'exécutant sur le *DM* de l'utilisateur, envoie un message « *propose* » (proposition de connexion) à l'*agent contrôleur de connexions*. S'il n'y pas d'*agent proxy* pour représenter cet *agent de DM*, l'*agent contrôleur de connexions* crée un *agent proxy* et envoie un message « *subscribe* » à l'*agent coordinateur* afin que l'*agent proxy* soit inscrit dans le système. L'*agent coordinateur* informe l'*agent de profil de DM* de cette inscription. L'*agent contrôleur de connexion* envoie également à l'*agent de DM* un message « *confirm* » lorsque le processus d'inscription a abouti (correctement ou non) (cf. Figure 7.1).

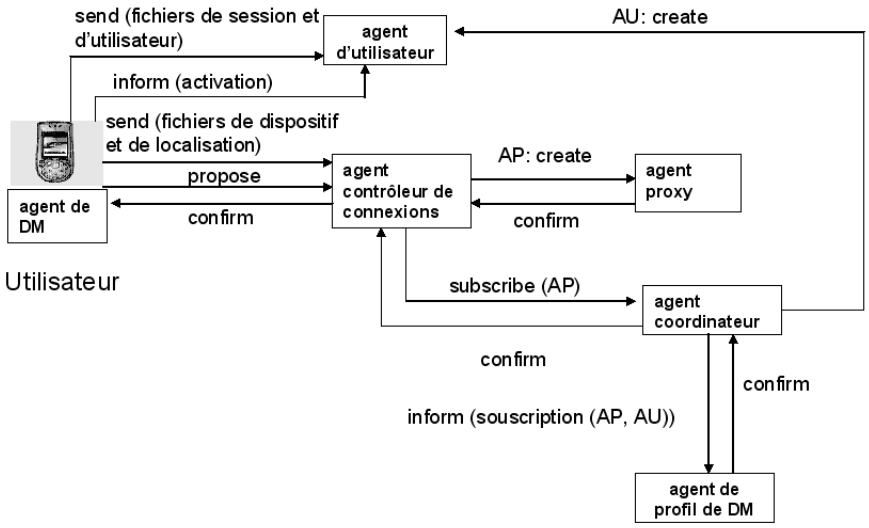


Figure 7.1. Scénario associé à la connexion.

Lorsque l'*agent de DM* reçoit le message de confirmation, un *agent d'utilisateur* est créé dans la plate-forme centrale de *PUMAS* afin de gérer le profil de l'utilisateur. Ce profil est défini dans le fichier *de session* en cours, géré par l'*agent de DM* et envoyé à l'*agent d'utilisateur*. L'*agent de DM* envoie aussi à l'*agent d'utilisateur* un fichier *d'utilisateur* contenant les préférences de l'utilisateur pour cette session. L'*agent de DM* envoie deux fichiers à l'*agent contrôleur de connexions* afin de faire connaître les caractéristiques du *DM* et celles de sa localisation (les fichiers respectivement *de dispositif* et *de localisation*).

7.3.2 Scénario d'envoi d'une requête d'information

Tout d'abord, le *DM* doit être connecté au système. S'il s'agit de la première connexion de l'utilisateur, l'*agent de DM* suit le même processus que celui décrit dans le scénario précédent. Si l'utilisateur a déjà été connecté, l'*agent de DM* suit le même processus de connexion que dans le scénario précédent excepté pour la partie d'envoi de fichiers (l'*agent de DM* a déjà envoyé les fichiers lors des sessions précédentes). L'*agent de DM* envoie ces fichiers dans le cas de l'existence de changements à notifier. Par exemple, lorsque l'utilisateur a changé de localisation (il envoie alors le fichier *de localisation*), ou de *DM* (il envoie alors le fichier *de dispositif*) ou lorsque l'utilisateur souhaite exprimer des préférences spécifiques pour la session en cours (il envoie le fichier *d'utilisateur*).

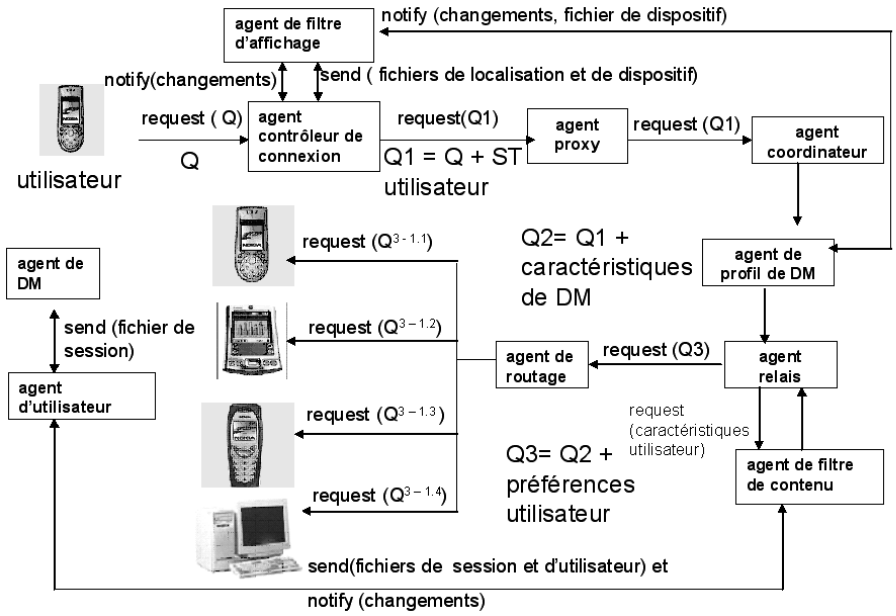


Figure 7.2. Scénario d'envoi d'une requête.

Lorsqu'un utilisateur envoie une requête d'information *Q* (cf. Figure 7.2), l'*agent de DM* la transmet à l'*agent contrôleur de connexions.* Si cet agent a reçu de nouveaux fichiers de *localisation* et de *dispositif*, il les envoie à l'*agent de filtre d'affichage*. Si l'utilisateur a établi comme préférence (définie dans le fichier *d'utilisateur*) le fait que sa requête *Q* dépende à la fois de sa localisation et du temps de connexion, l'*agent contrôleur de connexions* ajoute à *Q* l'information sur le temps de connexion, la localisation de l'utilisateur et les caractéristiques de connexion du *DM* de l'utilisateur. Cela conduit à la création d'une nouvelle requête *Q1* (voir Figure 7.2), définie par *Q1 = Q* + caractéristiques *Spatio-Temporelles* (*ST*) *de l'utilisateur*. Sinon, l'*agent contrôleur de connexions* ajoute uniquement à *Q* les caractéristiques de connexion du *DM* de l'utilisateur (à partir des données extraites du fichier *de dispositif*). La requête *Q1* est ensuite envoyée à l'*agent proxy*. *Q1* transite par l'*agent coordinateur*, puis par l'*agent de profil de DM*. Cet agent reçoit aussi de l'*agent de filtre d'affichage*, le fichier de *dispositif*, si ce dernier a reçu un nouveau fichier de *dispositif* ou des notifications sur des changements du dispositif. L'*agent de profil de DM* ajoute à *Q1* des caractéristiques liées au *DM*. Ces caractéristiques sont fournies par l'*agent de filtre d'affichage* qui les a déduites des requêtes précédentes ou les a extraites de sa *base de connaissances*. La nouvelle requête *Q2* (voir Figure 7.2), définie par *Q2 = Q1 + caractéristiques de DM* est envoyée par l'*agent de profil de DM* à un *agent relais.* L'*agent relais* ajoute à *Q2* les caractéristiques spécifiques de l'utilisateur dans le système. Ces caractéristiques sont demandées à l'*agent de filtre de contenu* (voir Figure 7.2). *Q3* est définie par *Q3 = Q2 + préférences de l'utilisateur*. L'*agent de DM* envoie à l'*agent d'utilisateur* un nouveau fichier *d'utilisateur* (contenant ses préférences) dans le cas où l'utilisateur exprime de nouvelles préférences pour la session en cours. L'*agent d'utilisateur* communique les nouvelles préférences à l'*agent de filtre de contenu* afin de générer un profil unique pour cet utilisateur, en donnant priorité aux nouvelles préférences pour

cette session. L'*agent relais* transmet *Q3* à l'*agent de routage* (chargé d'accomplir le processus de *routage de requêtes*, voir section 7) qui décide quels sont les *agents de SI* capables d'y répondre. Il peut envoyer la requête à un *agent de SI* spécifique ou à plusieurs *agents de SI* ou il peut décomposer la requête (voir section 7.1.1) en sous requêtes envoyées à un ou plusieurs *agents de SI*. La Figure 7.2 montre un scénario dans lequel *Q3* est décomposée en $Q^{3-1.1}$, $Q^{3-1.2}$, $Q^{3-1.3}$ et $Q^{3-1.4}$ qui sont envoyées aux *agents de SI* s'exécutant sur un serveur et différents *DM*.

Lorsqu'un utilisateur U_1 formule une requête d'information à destination d'un autre utilisateur U_2, tous deux étant équipés de *DM*, la requête est propagée de l'*agent de DM* s'exécutant sur le *DM* d'U_1 vers l'*agent de routage* de *PUMAS* qui la redirige à l'*agent de DM* s'exécutant sur le *DM* d'U_2. L'*agent du DM* d'U_2 change donc de rôle et devient un *agent de SI*, c'est-à-dire, l'agent en charge de répondre à la requête. Ce changement de rôle est possible car un *agent de DM* possède la connaissance pour gérer l'information stockée dans le *DM* sur lequel il s'exécute. De plus, l'*agent de DM* possède la capacité de répondre aux requêtes d'information.

7.3.3 Scénario de réception des résultats d'une requête d'information

Lorsque l'*agent de routage* reçoit tous les résultats de la requête des *agents de SI* (dans l'exemple de la Figure 7.3, l'*agent de routage* reçoit les résultats partiels – $RR^{1.1}$, $RR^{1.2}$, $RR^{1.3}$ et $RR^{1.4}$ – envoyés par les *agents de SI* s'exécutant sur un serveur et différents *DM*), il les analyse avant d'envoyer un message de « *confirms* » ou de « *disconfirms* » ou de « *not understand* » à l'*agent relais.* Ce message inclut les résultats de la requête (*RR*). L'*agent de filtre de contenu* envoie le fichier *d'utilisateur* à l'*agent relais* lorsque des changements sur les préférences de l'utilisateur sont survenus. L'*agent relais* vérifie que les résultats peuvent satisfaire les préférences de l'utilisateur (dans l'exemple de la Figure 7.3), *RR1* est le résultat de l'application du *filtre de contenu* à *RR* en tenant compte des préférences et de l'historique de l'utilisateur. Il les redirige à l'*agent de profil de DM*. L'*agent de filtre d'affichage* envoie le fichier *de dispositif* à l'*agent de profil de DM* lorsque des changements des caractéristiques du *DM* sont survenus. Cet agent vérifie si les résultats peuvent être affichés en tenant compte des caractéristiques du *DM* et il procède à la première étape du *filtre d'affichage* (dans l'exemple de la Figure 7.3, *RR2* est produit par le filtre de *RR1* en considérant les caractéristiques du *DM*). Ensuite, *RR2* est transmis par l'*agent coordinateur* à l'*agent proxy* et puis à l'*agent contrôleur de connexions*. L'*agent de DM* envoie le fichier *de localisation* à l'*agent contrôleur de connexions* lorsque des changements de la localisation de l'utilisateur ou des caractéristiques de connexion du *DM* sont survenus. L'*agent contrôleur de connexions* procède à la dernière étape du *filtre d'affichage* en tenant compte des caractéristiques de connexion du *DM* de l'utilisateur (si l'utilisateur est encore connecté, a changé de localisation, si un délai (« *timeout* ») est dépassé, *etc*.). Grâce aux filtres de *contenu* et d'*affichage*, les résultats de la requête reçus par l'*agent de DM* sont affichés sur le *DM* de l'utilisateur. Ces résultats correspondent à l'information la plus adaptée au contexte d'utilisation, aux préférences de l'utilisateur, aux caractéristiques du *DM* et à la requête.

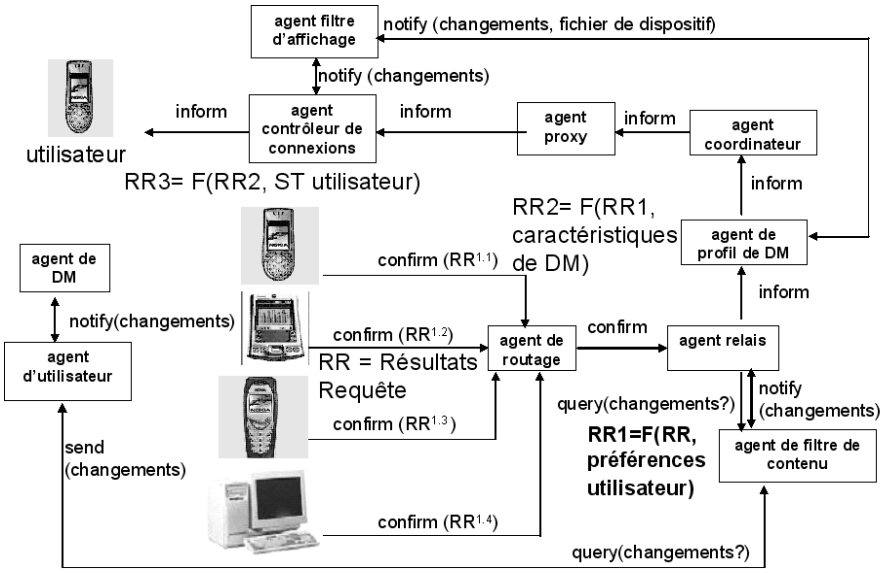


Figure 7.3. Scénario de réception des résultats d'une requête.

Il est important de noter que le scénario décrit ci-dessus inclut plusieurs étapes de vérification de résultats qui peuvent sembler inutiles puisque le scénario d'envoi d'une requête a déjà permis de raffiner la requête en tenant compte des caractéristiques de l'utilisateur et de celles de son *DM*. Cependant, cette information supplémentaire, ajoutée durant le scénario d'envoi d'une requête, pourrait ne plus être valide au moment de la livraison des résultats. Notamment, les caractéristiques de l'utilisateur (changements de localisation et de préférences) et les moyens de connexion et de communication (variation de bande passante, différents *DM* utilisés, problèmes de réseau) pourraient avoir évolués et, en conséquence, pourraient avoir un impact sur les résultats attendus par l'utilisateur. Les contrôles présentés dans le scénario « *réception des résultats d'une requête d'information* » visent à éliminer la transmission d'information non pertinente à l'utilisateur.

Dans la section suivante, nous illustrons le processus réalisé par les agents de *PUMAS* en utilisant l'exemple d'un *SIW* d'un hôpital auquel un médecin accède pour consulter les analyses médicales, les médicaments prescrits et le régime alimentaire d'un patient.

7.3.4 Exemple

Dans cet exemple, nous considérons que les *SI* d'un hôpital sont distribués entre plusieurs *DM* ou plusieurs *serveurs* (cf. Figure 7.4). Ils peuvent être accédés par les médecins à l'aide de *DM* (par exemple, des *PDA*). Les médecins peuvent donc recevoir l'information en fonction de leur localisation, leurs préférences, des caractéristiques techniques de leurs *DM* et de considérations sur le moment de la connexion. Ainsi, lorsque les médecins visitent un patient, ils peuvent à travers leur *DM* consulter le dossier médical du patient, ses analyses médicales, ses médicaments prescrits, *etc*. En indiquant la localisation du patient (par exemple, en donnant la chambre, le lit) et la date courante (extraite du système), l'identité du patient peut être connue grâce au système, et le médecin peut pour la suite obtenir les informations le concernant.

L'application s'exécute sur le *DM* d'un médecin et interagit avec différents *SI* de l'hôpital. Le médecin peut également communiquer directement avec d'autres médecins à travers son *DM*, afin d'obtenir leurs avis ou leur aide (par exemple, poser des questions au médecin spécialiste qui a précédemment examiné ce patient et qui lui a prescrit une analyse médicale).

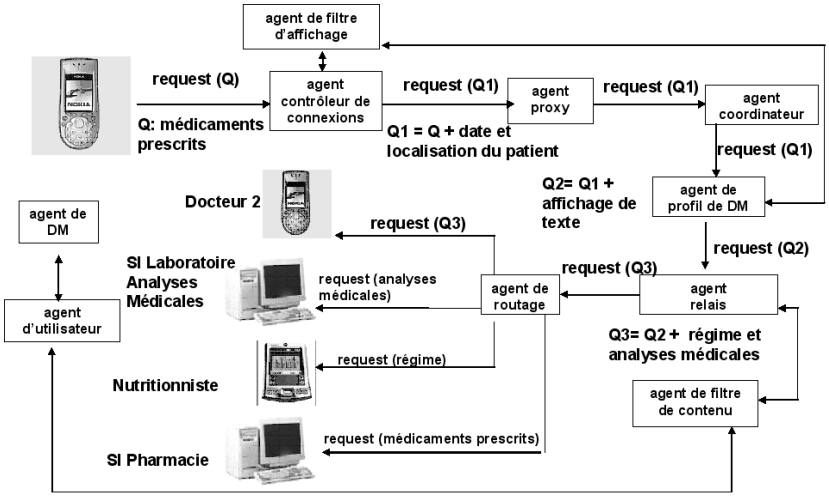


Figure 7.4. Envoi d'une requête dans le SI de l'hôpital.

Lorsqu'un médecin est dans la chambre d'un patient, il entre l'information sur la localisation du patient tandis que l'application obtient la date du système (information sur le moment de connexion). L'*agent de DM* (s'exécutant sur le *DM* du médecin) envoie la requête. La requête est propagée à travers le noyau de *PUMAS* : elle est tout d'abord transmise à l'*agent contrôleur de connexions*, puis aux *agents du SMA de communication* (*agents proxy*, *agent coordinateur* et *agent de profil de DM*). L'*agent de profil de DM* inclut dans la requête, l'information sur le *DM*. Par exemple, si le médecin est connecté à travers un *Palm Tungsten C*, ne supportant pas le format graphique, les réponses aux requêtes comme les résultats des analyses médicales ne pourront s'afficher que sous format texte. L'*agent de profil de DM* demande à l'*agent de filtre d'affichage* l'information sur ce *DM*. L'*agent de profil de DM* pourrait recevoir de l'*agent de filtre d'affichage* des faits définis de la manière suivante (conformément à la définition donnée à la section 6.2.4.2c –) :

(*deffacts CaractéristiqueDM*
(*TypeDM "Palm Tungsten C"*)
(*caractéristique* (*type "vidéo_non_supportée"*)(*description "réseau_Wi-Fi"*))
(*caractéristique* (*type "plusieurs_images"*)(*description "réseau_Wi-Fi"*)
(*caractéristique* (*type "texte"*) (*description "réseau_Wi-Fi" "Bluetooth"*)))

Ensuite, l'*agent de profil de DM* envoie la requête à l'*agent relais* attachant les préférences d'information précédemment exprimées par le médecin. Ces préférences sont traduites comme faits par *l'agent d'utilisateur* et l'*agent de filtre de contenu*, en ajoutant aux préférences d'information, les actions qui expriment la manière dont le système doit réagir afin d'afficher l'information sur le dispositif d'accès et/ou de résoudre des problèmes.

Par exemple, un médecin exprime ses préférences d'information à travers une interface de saisie. Ces préférences sont traduites par l'*agent d'utilisateur* comme décrit ci-dessous. Un exemple de préférence d'information est la suivante[76] : « *chaque fois que je demande une analyse médicale, le système devra aussi me fournir le régime du patient et les médicaments prescrits ; je préfère les résultats sous un format graphique mais si mon DM ne supporte pas ce format, j'aimerais les recevoir sous un format de texte* ». Cette préférence peut être traduite par le fait suivant (stocké dans la *base de connaissances* de l'*agent d'utilisateur*) :

(1) (*deffacts Préférence_Information_augmentée*
(2) (*utilisateurID "Docteur Thierry Dupont"*)
(3) (*info_requise "analyses médicales"*)
(4) (*info_complémentaire "régime" "médicaments prescrits"*)
(5) (*action*
(6) (*nom "montrer"*)
(7) (*attribut* (*nom "ordre"*)
(8) (*items "analyses médicales" "régime" "médicaments prescrits"*))
(9) (*attribut* (*nom "format_graphique"*)
(10) (*items "JPEG"*))
(11) (*problème*
(12) (*nom "Multimédia Non Supporté Par DM"*)
(13) (*type "incompatibilité"*)
(14) (*causes "Seulement Fichiers Texte Supportés"*))
(15) (*action*
(16) (*nom "montrer"*)
(17) (*attribut* (*nom "ordre"*)
(18) (*items "analyses médicales" "médicaments prescrits" "régime"*))
(19) (*attribut* (*nom "format_texte"*) (*items "XML" "txt"*)))

Les lignes (1) à (4) correspondent au fait d'une *préférence d'information* simple (l'info_requise correspond aux analyses médicales et l'info_complémentaire correspond au régime et aux médicaments prescrits d'un patient). Les lignes (5) à (10) concernent la manière (*action*) dont le système doit réagir pour afficher l'information. Le système doit afficher dans un ordre spécifique l'information en utilisant un format graphique. De la ligne (11) à la ligne (14), on décrit un problème spécifique (pouvant survenir) et finalement les lignes (15) à (19) concernent la manière (*action*) dont le système doit réagir si ce problème survient. L'action correspond à la manière dont le système doit afficher l'information dans un ordre spécifique en utilisant un format texte.

L'*agent d'utilisateur* transfère cette information à l'*agent de filtre de contenu* stockant ce fait et l'envoie à l'*agent relais*. L'*agent relais* ajoute cette préférence à la requête et l'envoie à l'*agent de routage*. L'*agent de routage* reçoit la requête augmentée, et, grâce à l'information sur le *SI*, il peut la décomposer en sous-requêtes et ensuite les rediriger chacune vers le *SI* approprié (cf. section 7.1.1). Les faits définis dans la section 7.1.1 sont exploités dans cet exemple par l'*agent de routage* afin de rediriger les requêtes aux *agents de SI* des *SI* de l'hôpital.

L'*agent de routage* redirige la requête à l'*agent de SI* s'exécutant sur le *SI* qui gère l'information sur les patients dans l'hôpital. Toutes les requêtes suivent le même chemin de

[76] Un traitement de préférences plus détaillé sera présenté dans la section 8.1

l'*agent de DM* vers l'*agent de routage*. Si le médecin souhaite connaître les derniers *médicaments prescrits* à ce patient, l'*agent de routage* redirige la requête à l'*agent de SI* s'exécutant sur le *SI de la Pharmacie*. Si la requête concerne un autre *médecin* (*pair*), l'*agent de routage* redirige la requête à l'*agent de SI* s'exécutant sur le *DM* du pair. Un médecin peut aussi demander l'information sur un patient spécifique à plusieurs de ses pairs. Dans ce cas, l'*agent de routage* pourrait envoyer la requête par « *broadcast* » ou il pourrait décomposer la requête en tenant compte du pair récepteur (par exemple, les requêtes liées au cœur pour le cardiologue) ou en prenant en compte les critères définis dans le fichier *d'utilisateur* (par exemple, si le critère d'adaptation de la requête est la *localisation*, les requêtes doivent seulement être redirigées aux médecins dans la même ou dans une localisation proche de l'expéditeur de la requête). L'information récupérée est organisée par l'*agent de routage* (par exemple, les derniers médicaments prescrits, les réponses de médecins pairs sur ce patient) et est retournée au médecin qui a envoyé la requête en suivant le chemin inverse. Les différents agents doivent vérifier les résultats car, par exemple, le médecin peut s'être déconnecté du système (par exemple, à cause de problèmes de réseau), et récupéré sa session dans une nouvelle connexion dont les caractéristiques sont différentes de celles de la précédente : L'utilisateur peut maintenant consulter le système en utilisant un autre type de *DM* qui supporte des formats graphiques (constituant une préférence du médecin qui peut maintenant être satisfaite).

7.4. Conclusion

Lorsqu'un utilisateur formule des requêtes, les résultats peuvent provenir de différents *Systèmes d'Information* (*SI*). Dans ce chapitre, nous avons défini un processus de « *routage de requêtes* » comme un mécanisme qui analyse la requête en identifiant ses items, et en faisant la correspondance (sémantique ou syntaxique) entre les items et l'information gérée par les *SI* connus par l'*agent de routage*, afin de sélectionner le(s) *SI* capable(s) d'y répondre. Après l'identification de ces items et la reconnaissance des *SI* qui peuvent gérer des items équivalents, ce processus décompose la requête. Un *agent de routage* tient compte des critères d'adaptation fournis par l'utilisateur (tels que la localisation, les activités développées par l'utilisateur pendant une période, ses préférences, *etc*.) afin de choisir les *SI* les plus appropriés pour répondre à la requête. Finalement, ce processus doit compiler les résultats de la requête.

Nous avons décrit trois scénarios correspondants à la connexion d'un utilisateur à *PUMAS*, à l'envoi d'une requête, et à la réception des résultats. Nous avons terminé ce chapitre par la présentation d'un exemple impliquant les différents *SI* d'un hôpital dans lequel un médecin demande des informations sur les prescriptions, les analyses médicales et le régime alimentaire d'un patient.

Bien que nous ayons utilisé ici une représentation des préférences d'information de l'utilisateur, ce chapitre n'a pas abordé la représentation de ses préférences concernant les activités qu'il souhaite accomplir dans le système, les résultats attendus de ces activités et la manière dont l'utilisateur souhaite que les résultats de la requête soient affichés sur son dispositif d'accès. Les *préférences d'affichage* sont aussi un composant de base pour le *filtre d'affichage* (qui adapte l'information en considérant les caractéristiques du dispositif d'accès de l'utilisateur). Le

chapitre suivant est consacré à la représentation des différents types de préférences de l'utilisateur et à la génération de son profil qui, confronté aux caractéristiques contextuelles de la session courante, permettra l'adaptation.

8. GESTION DES PREFERENCES DE L'UTILISATEUR

Afin d'adapter l'information à l'utilisateur, il est nécessaire de tenir compte du *contexte d'utilisation*[77], c'est-à-dire des caractéristiques de la session en cours et de celles du dispositif. Les travaux menés au sein de notre équipe par Kirsch-Pinheiro [Kirs06] sur la représentation et la gestion du *contexte d'utilisation* ont constitué une première proposition en ce sens. Ils permettent de décrire des *contextes d'utilisation potentiels*, prédéfinis, auxquels le système sait réagir, c'est-à-dire sait s'adapter. L'approche consiste à appliquer un algorithme de comparaison visant à confronter la description du contexte courant d'utilisation à celles de ces contextes potentiels. Le contexte courant est assimilé à un contexte connu. L'appariement réalisé par le système permet d'appliquer un certain nombre d'actions préconisées pour adapter la réponse à ce contexte. Notre travail vise une adaptation du système qui tient compte du *contexte courant d'utilisation* mais adopte une approche complémentaire à cette approche dans la mesure où nous nous intéressons plus particulièrement à la description de l'ensemble des *préférences de l'utilisateur* comme élément du *contexte d'utilisation*. Ces préférences traduisent les souhaits de l'utilisateur à propos des activités qu'il a à mener, des contenus que le système lui délivre et de l'affichage des informations. Leur prise en compte est évaluée au regard du contexte courant d'utilisation : en effet, toute préférence peut ne pas pouvoir être satisfaite si certaines conditions ne sont pas réunies. Par exemple, c'est le cas si l'utilisateur exprime par une préférence qu'il souhaite privilégier les données vidéo mais que le dispositif utilisé lors d'une session particulière ne supporte pas ce type de données. Ceci requiert de disposer d'outils permettant de représenter les profils des utilisateurs et de mécanismes pour exploiter ces représentations en vue de l'adaptation des *Systèmes d'Information basés sur le Web* (*SIW*). Nous nous intéressons ici plus particulièrement aux outils de représentation des préférences et aux mécanismes dédiés. De plus, nous abordons la question de l'adaptation (basée sur les préférences) de *SIW* auxquels on accède depuis des *Dispositifs Mobiles* (*DM*).

[77] Dans notre proposition, le *contexte d'utilisation* est constitué d'information sur la localisation de l'utilisateur, les caractéristiques du *DM*, les droits d'accès de l'utilisateur et ses activités dans le système.

Dans ce chapitre nous présentons la classification de *préférences de l'utilisateur* que nous avons établie et qui nous a conduits à distinguer des préférences d'*activité*, de *résultat* et d'*affichage*. Puis, nous montrons comment le *profil contextuel* d'un utilisateur est généré à l'aide de l'algorithme de *correspondance contextuelle*, à partir des *préférences de l'utilisateur* et du *contexte d'utilisation* de la session en cours. Nous décrivons la résolution des conflits qui peuvent apparaître entre les *préférences de l'utilisateur* constitutives du *profil contextuel* obtenu. Nous avons défini un *Système de Gestion de Profils Contextuels* (*SGPC*) qui fournit les éléments permettant de procéder à l'adaptation de l'information, compte tenu des *préférences de l'utilisateur* et du *contexte d'utilisation*. Ensuite, nous proposons un exemple visant à expliquer les étapes de la représentation des préférences et de la génération du *profil contextuel*. Enfin, nous intégrons les processus de gestion de préférences ainsi que celui de génération de profils contextuels dans *PUMAS*.

8.1. Formalisation

Nous distinguons trois types de préférences :

1. Les *préférences d'activité* qui concernent les activités qu'un utilisateur souhaite accomplir dans le système. Du côté du système, une *activité* consiste en un ensemble de *fonctionnalités* (c'est-à-dire, l'invocation d'un service, ou l'exécution d'une application) qui doivent être exécutées afin d'accomplir cette *activité*. Une *fonctionnalité* peut être liée à d'autres fonctionnalités qui s'exécutent de manière séquentielle, concurrente ou conditionnelle par rapport à celle-ci. Nous nous limitons ici aux *fonctionnalités de consultation* telles que définies par Villanova-Oliver [Vill02] qui permettent de consulter des informations au moyen de requêtes. Ces requêtes sont donc associées à ces fonctionnalités, et en quelque sorte, les rendent opérationnelles.
2. Les *préférences de résultat* qui concernent le *contenu* : l'utilisateur peut choisir les résultats à lui délivrer parmi ceux obtenus après l'exécution des fonctionnalités et déterminer leur ordre de présentation[78].
3. Les *préférences d'affichage* concernent la manière dont l'utilisateur souhaite que l'information soit affichée sur son *DM*. Ceci recouvre d'un côté l'apparence, le style, les polices de caractères, *etc.*, et, de l'autre, les caractéristiques des formats d'affichage (c'est-à-dire, les caractéristiques de la vidéo, des images, ou du son).

À l'intérieur de chaque type de préférence, nous distinguons les *préférences générales* (qui s'appliquent à toutes les sessions ; elles constituent la valeur par défaut) et les *préférences spécifiques* (qui s'appliquent à la session en cours). Une *session* débute lorsqu'un utilisateur se connecte au système et exécute une ou plusieurs *fonctionnalités* offertes. Nous désignons par *F* une *fonctionnalité* de nom *nomf* définie comme un tuple composé d'une liste de paramètres d'entrée ($<e_1, e_2 \ldots e_n>$) et d'une liste de paramètres de sortie ($<s_1, s_2 \ldots s_k>$) :

$$F = nomf(<e_1, e_2 \ldots e_n>, <s_1, s_2 \ldots s_k>)$$

[78] Les *préférences de résultats*, présentées ici dans leur version la plus simple, peuvent prendre une forme offrant encore plus de flexibilité et reposant sur les principes de l'accès progressif décrits par Villanova-Oliver [Vill02].

Afin d'illustrer la notion de *fonctionnalité*, nous reprenons l'exemple d'un *utilisateur-médecin* qui accède à l'aide de son *DM* au *Système d'Information* (*SI*) de l'hôpital dans lequel il exerce. Lorsqu'il ouvre une session, supposons qu'il exécute l'*activité* « *Consultation du dossier d'un patient* ». Cette *activité* est accomplie par le système à travers l'invocation de deux fonctionnalités : « *Consulter les analyses médicales* » et « *Consulter les médicaments prescrits* ». Ces deux fonctionnalités ont comme paramètre d'entrée « *le nom du patient* » et comme résultats respectifs « *les analyses médicales* » et « *les médicaments prescrits* » de ce patient.

Une modélisation de chacun des trois types de préférence (*activité*, *résultat* et *affichage*) est présentée et illustrée dans ce qui suit.

8.1.1 Préférences d'activité

Les *préférences d'activité* décrivent la manière dont l'utilisateur envisage d'accomplir ses activités dans le système. Nous définissons ce type de préférence de la manière suivante :

`Préférence_Activité(type, critères, A)`

où **`type`** prend pour valeur « *générale* » ou « *spécifique* », **`critères`** est un ensemble de critères d'adaptation (par exemple, la localisation et le type de *DM*) dont le système tient compte pour l'exécution des fonctionnalités dans une session, et **`A`** est l'*activité* que l'utilisateur souhaite accomplir dans le système. Cette activité est exprimée à travers une *chaîne de fonctionnalités* exécutées de manière séquentielle, concurrente ou conditionnelle. Cette chaîne est décrite et régie par la grammaire suivante, définie en notation *BNF* (« *Backus Naur Form* ») :

```
Activité::= fonctionnalité [op fonctionnalité] |conditionnelle | boucle | nil ;
op::= séquence|concurrent ;
séquence ::= ";";
concurrent ::= "|" ;
conditionnelle ::= "if" <condition> "then" Activité ["else" Activité] "end if" ;
boucle ::= "while" <condition> "do" Activité "end while" ;
```

Nous donnons trois cas illustrant les différentes possibilités d'expression de *préférences d'activité.*

Le premier cas consiste à exécuter une activité composée seulement d'une *fonctionnalité*. Par exemple, afin d'exécuter la fonctionnalité F_i « *Consulter une analyse* », un utilisateur-médecin doit fournir comme paramètres d'entrée le *nom du patient* et le *nom de l'analyse :*

F_i`=Consulter_une_analyse(<nom_patient,nom_analyse>,<résultats_analyse>)`

Une *préférence d'activité* pour F_i, générale et sans critère d'adaptation, est définie par :

`Préférence_Activité (générale, (),` F_i`)`

Il est possible d'associer une *préférence de résultat* à toute fonctionnalité F_i afin de choisir les résultats souhaités de cette fonctionnalité :

`Préférence_Activité (générale, (),` (F_i, $PrefResF_i$)`)`

où $PrefResF_i$ est une *préférence de résultat* telle que définie dans la section 8.1.2. Dans les exemples suivants, les fonctionnalités ne sont pas associées à de telles préférences par souci de clarté.

Dans le deuxième cas, nous considérons qu'un utilisateur souhaite exécuter une activité *A* composée de plusieurs fonctionnalités F_j (j ∈ [1..n]).

Par exemple, chaque fois qu'un médecin a un rendez-vous avec un patient, le médecin exécute la fonctionnalité « *Débuter rendez-vous* ». En même temps, le système déclenche (comme cela est défini dans la *préférence d'activité* du médecin) les fonctionnalités « *Consulter_dossier* » sur le dossier de ce patient, « *Consulter_heure* » pour activer une horloge afin de contrôler la durée de ce rendez-vous, et « *Consulter_agenda* » pour consulter son agenda dans le but de fixer son prochain rendez-vous avec ce patient :

F_1=DebuterRendez-Vous(<nom_patient>, <dossier>)
F_2=Consulter_heure(<>, <hh:mm:ss>)
F_3=Consulter_dossier(<nom_patient>, <dossier>)
F_4=Consulter_agenda(<>, <hh:mm:ss>)
Préférence_Activité (générale, (), F_1;(F_2|F_3); F_4)

Un cas particulier peut survenir lorsque l'utilisateur, à chaque exécution d'une *fonctionnalité F*, exécute fréquemment d'autres fonctionnalités que l'on peut alors considérer comme associées. L'exécution de ces *fonctionnalités* peut être anticipée par le système si un historique d'*activités* (composé de l'ensemble des *activités* qu'il a accomplies lors des sessions précédentes) est maintenu. Par exemple, chaque fois qu'un utilisateur-médecin exécute la fonctionnalité « *Consulter_analyse_médicale* », le système (basé sur l'historique de l'utilisateur) remarque qu'il exécute fréquemment les fonctionnalités « *Consulter_régime* » et « *Consulter_médicament* » et génère alors une *préférence d'activité* traduisant cette observation.

F_1=Consulter_analyse_médicale(<nom_patient>,<résultats_an_méd>)
F_2=Consulter_régime(<nom_patient>, <régime>)
F_3=Consulter_médicaments(<nom_patient>, <médicaments_prescrits>)
Préférence_Activité (générale, (), F_1;F_2;F_3)

Le troisième cas consiste à exécuter une ou plusieurs fonctionnalités en tenant compte de critères (tels que sa localisation, son *DM*, *etc*.) définis par l'utilisateur. Les résultats délivrés par une fonctionnalité peuvent être différents selon les valeurs de ces critères. Par exemple, lorsqu'un *utilisateur-médecin* consulte *son emploi du temps* à l'*hôpital*, il obtient comme résultat *les opérations chirurgicales* auxquelles il participe *cette semaine*, mais lorsqu'il le consulte de son *cabinet*, il obtient comme résultat ses *consultations* avec ses patients. La *préférence* et les *fonctionnalités* sont définies comme suit :

$F_{1=}$ Consulter_chirurgies(<nom_médecin, date>, <chirurgies>)
$F_{2=}$ Consulter_consultations(<nom_médecin, date>, <consultations>)
Préférence_Activité (générale,(localisation),{if (localisation == ″hôpital″) then F_1 else (if (localisation ==″cabinet″) then F_2})

8.1.2 Préférences de résultat

Les *préférences de résultat* permettent à l'utilisateur de fixer et d'ordonner les contenus délivrés suite à l'exécution d'une fonctionnalité. On définit :

Préférence_Résultat (type, *F*, <(s_1,*PAff*$_1$),(s_2, *PAff*$_2$)...(s_k, *PAff*$_k$)>)

où ***F*** est le nom de la fonctionnalité et le dernier terme est une liste ordonnée[79] de couples (s_i, $PAff_i$) où s_i représente un résultat de *F* et $PAff_i$ la *préférence d'affichage* (cf. section 8.1.3) qui s'applique à ce résultat. Un utilisateur peut ainsi choisir, parmi les résultats délivrés par la fonctionnalité, ceux qu'il souhaite obtenir (par exemple seulement s_3, s_5, s_8). Dans ce cas, la *préférence de résultat* définie est :

$$\text{Préférence\_Résultat (type, } F, < (s_3, PAff_1),(s_5, nil),(s_8, PAff_3)>)$$

où *nil* signifie que l'utilisateur n'a pas défini de préférence pour afficher ce résultat.

Par exemple, un médecin souhaite pour « *Consulter les analyses médicales* », ne recevoir que les résultats de la *courbe de glycémie* (sous forme d'image avec les caractéristiques de la *préférence d'affichage* $PAff\_image_1$) et les *analyses médicales* liées à la *thyroïde* (définies par *T3*, *T4* en format texte, respectivement associé aux *préférences d'affichage* $PAff\_texte_1$ et $PAff\_texte_2$). La représentation de la *préférence de résultat* est la suivante :

$$\text{Préférence\_Résultat(générale, "Consulter les analyses médicales",}$$
$$<(\text{courbe\_glycémie},PAff\_image_1),(T3,PAff\_texte_1),(T4,PAff\_texte_2)>)$$

8.1.3 Préférences d'affichage

Les *préférences d'affichage* décrivent la manière dont l'utilisateur souhaite que son *DM* affiche l'information (par exemple, au format image). Ces préférences sont définies par des tuples comme suit :

```
Preference_Affichage (type, format, {caractéristiques}, substitution).
```

Dans ces tuples, le ***type*** peut prendre pour valeur « *générale* » ou « *spécifique* », le ***format*** peut prendre pour valeur : « *vidéo* », « *texte* », « *image* » et « *audio* » et ***caractéristiques*** précise les valeurs prises par les attributs qui caractérisent le format. Le terme ***substitution*** correspond à une autre *préférence d'affichage* que le système tentera d'utiliser à la place de celle définie, si cette dernière ne peut pas être satisfaite (*substitution* peut valoir *nil*). Dans l'exemple ci-dessous, la *préférence d'affichage* P_1 correspond à une préférence pour l'affichage d'une vidéo, en donnant ses dimensions et le type de fichier (par exemple, hauteur, largeur, type) :

$$P_1 = \text{Preference\_Affichage (générale, vidéo,\{200,300, AVI\}, } P_2)$$

où P_2 est la préférence de substitution P_1 et contient les caractéristiques pour du texte (police, taille, couleur, type de fichier) :

$$P_2 = \text{Preference\_Affichage (générale, texte, \{Arial, 10, bleu, .doc\}, nil)}$$

Notons que les *préférences d'affichage* peuvent être référencées dans les *préférences de résultat*, constituant dans ce dernier cas, des préférences à appliquer à un format, indépendamment d'un contenu particulier (pour privilégier le format texte pour une session, par exemple).

[79] L'ordre de la liste exprime l'ordre de présentation des résultats. Le premier couple de la liste correspond au premier résultat présenté.

8.2. Définition du profil de l'utilisateur

Dans cette section, nous montrons comment le *profil contextuel* d'un utilisateur est généré à l'aide de l'algorithme de *correspondance contextuelle*, à partir du *profil utilisateur* et du *contexte d'utilisation* de la session en cours.

8.2.1 Vue d'ensemble du processus

Afin d'adapter l'information aux caractéristiques de l'utilisateur et à celles de son *DM*, nous adoptons un processus en deux étapes (cf. Figure 8.1). Une première étape de *Gestion de Préférences* (*étape 1*) est accomplie par le *Système de Gestion de Profils Contextuels*. Elle consiste en la capture du *contexte d'utilisation* de la session (c'est-à-dire, l'information sur les activités de l'utilisateur, les droits d'accès au système, la localisation, les caractéristiques du dispositif d'accès) et en la sélection des préférences (d'*activité, de résultat* et *d'affichage*) qui peuvent être appliquées, en considérant la session en cours (voir Figure 8.1, *Filtrage de Préférence*). Par exemple, pour la capture de la *localisation*, il est possible d'utiliser un dispositif *GPS* ou des méthodes telles que le *SNMP* (« *Simple Network Management Protocol* ») proposé par Nieto-Carvajal *et al.* [Niet04]. Le résultat de l'*étape 1* est un *profil contextuel de l'utilisateur* (composé des préférences sélectionnées, cf. section 8.2.2) et une ou plusieurs *requêtes* dites « *augmentées* ». Une requête augmentée correspond à la *requête initiale* (réalisation des fonctionnalités appartenant à une activité) à laquelle s'ajoutent les informations issues des *préférences de résultat* et *d'affichage*. Le profil contextuel de l'utilisateur et les requêtes augmentées constituent les entrées de l'étape de *Traitement de Requêtes* (étape 2 du processus, décrit section 7).

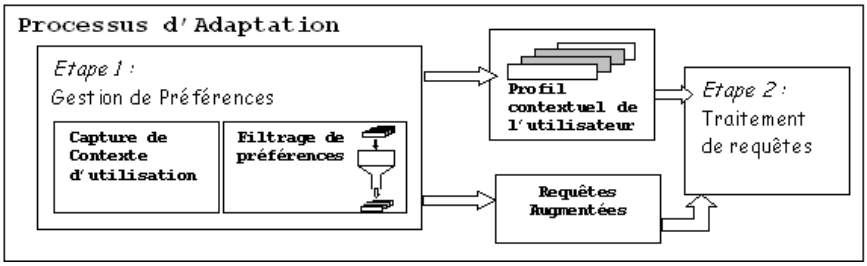


Figure 8.1. Processus d'adaptation.

Dans ce qui suit, nous détaillons l'étape de *Gestion de Préférences* prise en charge par le *Système de Gestion de Profils Contextuels* (*SGPC*).

8.2.2 Définition du profil contextuel de l'utilisateur

Nous définissons un « *profil utilisateur* » [80] comme un ensemble de descriptions, appelées *préférences de l'utilisateur*, qui ont trait :

[80] Dans notre proposition, l'expression « *profil utilisateur* » doit être comprise comme désignant ni plus ni moins qu'un ensemble des *préférences de l'utilisateur*. Nous rappelons en effet que nous ne nous intéressons pas ici à d'autres éléments pouvant entrer dans la constitution d'un profil.

- aux *activités* telles qu'un utilisateur peut souhaiter les planifier (orchestrer) lorsqu'il utilise le système, ce qui nécessite de dire quelles sont ces activités et comment elles sont organisées (par exemple, de façon séquentielle, concurrente, ou conditionnelle). Du point de vue du système, une *activité* correspond à un ensemble de *fonctionnalités* implémentées qui doit être exécuté afin d'accomplir cette activité ;
- au type et à l'ordre des résultats de ces activités (que nous appelons « *contenus* ») ;
- à la manière dont l'utilisateur souhaite que ces contenus soient affichés sur son *DM* (spécification du format attendu – de l'image, de la vidéo, du texte – selon les caractéristiques du *DM*).

Afin d'illustrer notre proposition, considérons un utilisateur qui souhaite recevoir sur son téléphone portable des prévisions météorologiques. Son *profil utilisateur* peut, par exemple, préciser que lorsqu'il exécute l'*activité* « *consultation des prévisions météo* », cet utilisateur est seulement intéressé par un *contenu* constitué des prévisions concernant la ville où il se trouve, uniquement pour la journée en cours, et avec des températures données en degrés Fahrenheit. Son profil peut également indiquer qu'il souhaite que l'*affichage* soit réalisé sous forme de carte plutôt qu'en mode textuel. De plus, notre proposition permet, en fonction du résultat retourné par une activité, de définir qu'une autre activité doit être engagée. Ainsi, lorsque les prévisions météo annoncent de la pluie (analyse du contenu), l'utilisateur souhaite que l'activité « *consulter les programmes des cinémas* » soit automatiquement engagée et paramétrée en fonction des données de son profil (concernant la localisation, l'heure, les chaînes de cinéma et le genre cinématographique préférés, *etc*.)

Dans notre proposition, le *profil contextuel de l'utilisateur* est composé des préférences *d'activité*, *de résultat* et *d'affichage* qui s'appliquent compte tenu du *contexte d'utilisation* d'une session (c'est-à-dire, compte tenu des caractéristiques de la session en cours, de celles du dispositif, *etc*.). Par exemple, on ne conserve, parmi les *préférences d'affichage*, que celles qui peuvent être satisfaites étant données les caractéristiques du dispositif d'accès, ou parmi les *préférences de résultat*, que celles qui ne sont pas en contradiction avec les droits d'accès de l'utilisateur.

Le *profil contextuel PC* (*u*, *s*, *DM*) = $\{P_1, P_2, P_3 \ldots P_k\}$ d'un utilisateur « *u* », qui se connecte à travers un dispositif « *DM* » lors d'une session « *s* » est l'ensemble des *k préférences de l'utilisateur* qu'il convient de retenir, compte tenu du *contexte d'utilisation*. Il est à noter que le *PC* est construit à partir de l'examen de l'ensemble $P_u = \{P_1, P_2, P_3, \ldots P_i, \ldots, P_n\}$ constitué de *toutes* les *préférences de l'utilisateur* définies pour l'utilisateur *u*, ce qui comprend les préférences définies indépendamment du *contexte d'utilisation* (préférences de type *général*) et celles définies pour la session en question (préférences de type *spécifique*).

L'examen de l'ensemble P_u, par l'algorithme décrit dans la section suivante, repose sur une phase préalable d'ordonnancement des préférences. Nous donnons les principes majeurs du système de priorités établi pour l'ordonnancement des préférences :

- tout d'abord sont examinées les *préférences d'activité*, puis celles de *résultat* et enfin celles *d'affichage ;*
- dans chaque classe de préférences (*activité, résultat, affichage*), les *préférences* spécifiques sont prioritaires sur les *préférences générales.* S'il n'y a pas de *préférences spécifiques* indiquées, le système ne considère que les *préférences générales.*

Si aucune préférence n'est définie, le système considère *l'historique de l'utilisateur* dans le *SI* et, en dernier ressort, les contraintes inhérentes à son *DM* (dans ce cas, le système construit lui-même des préférences).

8.2.3 Algorithme de correspondance contextuelle

Pour chacune des préférences P_i de l'ensemble complet des préférences P_u, nous appliquons l'algorithme suivant afin de vérifier si P_i peut être ajoutée au *profil contextuel* (cf. lignes 2 à 4 de l'algorithme). Le système analyse chaque préférence P_i (appartenant à P_u) et son ensemble des préférences de substitution[81]. Cette analyse consiste à évaluer, dans l'ordre, les trois cas suivants, et se termine avec le premier qui soit satisfait : *i)* si P_i peut être satisfaite (cf. ligne 7) alors P_i. s'ajoute au profil contextuel et sa *chaîne de préférences* de substitution est ajoutée à la *liste de préférences rejetées ; ii)* si une préférence de substitution de P_i peut être satisfaite (cf. lignes 18 à 37) alors cette préférence de substitution s'ajoute au profil contextuel, et le reste de la *chaîne de préférences* de substitution est ajouté à la *liste de préférences rejetées* (cf. lignes 13, 16, 26, 34) ; *iii)* si P_i n'a pas de préférences de substitution, rien n'est ajouté au profil contextuel ni à la *liste de préférences rejetées*.

L'algorithme précédent est présenté ci-dessous et nous utilisons les abréviations suivantes : **PA** est la *Préférence Analysée*, **LPR** est la *Liste de Préférences Rejetées*, **NPA** est la *Nouvelle Préférence Analysée* et **PC** est le *Profil Contextuel*.

[81] Les *préférences de substitution* ne sont définies que pour les *préférences d'affichage*.

```
(1)i←1
(2)Tant que (i <= n) faire // Pour chaque Pi où i ∈ [1,n] :
(3)   PA ← Pi    //Préférence qui est analysée dans cette itération
(4)   si (Pi ∈ PC ou Pi ∈ LPR) alors
(5)      skip  //Le système analysera Pi+1 (c'est-à-dire PA = Pi+1)
(6)   sinon
(7)      si (Pi peut être satisfaite) alors
(8)        Ajoute Pi dans PC
(9)        PA←préférence de substitution de Pi
(10)       Tant que (PA <> nil) faire
(11)        NPA← PA;
(12)        PA← Préférence de substitution de NPA
(13)        Ajoute NPA dans la LPR si et seulement si NPA∉PC et NPA∉LPR
(14)       fin Tant que
(15)     sinon
(16)       Ajoute Pi dans la LPR si et seulement si Pi∉PC et Pi∉LPR
(17)       PA←préférence de substitution de Pi
(18)       Tant que (PA <> nil) faire
(19)         si (PA peut être satisfaite et PA∉PC et PA∉LPR) alors
(20)           NPA ← PA;
(21)           PA←préférence de substitution de la NPA
(22)           Ajoute NPA dans PC si et seulement si NPA∉PC et NPA∉LPR
(23)           Tant que (PA <> nil) alors
(24)             NPA ← PA;
(25)             PA ← préférence de substitution de la NPA
(26)             Ajoute NPA dans LPR si et seulement si NPA∉PC et NPA∉LPR
(27)           fin Tant que
(28)         sinon
(29)           si (PA∈PC ou PA∈LPR) alors
(30)             PA ← préférence de substitution de PA
(31)           sinon
(32)             NPA ← PA;
(33)             PA ← préférence de substitution de NPA
(34)             Ajoute NPA dans LPR si et seulement si NPA∉PC et NPA∉LPR
(35)           fin si
(36)         fin si
(37)       fin tant que
(38)     fin si
(39)  fin si
(40)  i ← i + 1. // Incrémenter i
(41) Fin Tant que
```

L'algorithme peut rejeter des préférences de substitution qui pourront devenir utiles lors de l'adaptation de l'information. Par exemple, supposons qu'une préférence « *a* » soit la préférence de substitution des préférences « *b* » et « *c* ». Si « *b* » est satisfaite, « *a* » sera ajoutée à la *liste de préférences rejetées*. Dans le cas où « *c* » ne peut pas être satisfaite, « *a* » devrait être de nouveau analysée afin de déterminer si elle peut être satisfaite. D'ailleurs, certaines préférences rejetées peuvent décrire les caractéristiques d'un format de media qui doit être affiché. Par exemple, supposons que « *a* » soit la préférence de substitution de « *b* ». « *a* » spécifie des caractéristiques associées à du texte et « *b* » spécifie des caractéristiques associées à la vidéo. Si la vidéo est supportée par le *DM*, le système ajoute « *a* » (les caractéristiques de texte) à la *liste de préférences rejetées* privilégiant ainsi la *préférence de l'utilisateur* pour la vidéo. S'il existe cependant une information qui ne peut être affichée qu'en format texte, le système est capable de retrouver dans la *liste de préférences rejetées* les caractéristiques préférées par l'utilisateur pour le texte (« *a* ») et peut ainsi les

appliquer. Pour ces raisons nous gérons une *liste de préférences rejetées* afin de disposer de préférences rejetées précédemment.

Afin d'illustrer l'algorithme de génération du *profil contextuel de l'utilisateur*, nous supposons que l'ensemble P_u des préférences de cet utilisateur est celui présenté dans la Figure 8.2. Si P_1 est satisfaite, P_5, P_2 et P_4 sont ajoutées à la *liste de préférences rejetées*. Les *profils contextuels* générés par l'algorithme sont présentés dans le Tableau 6.

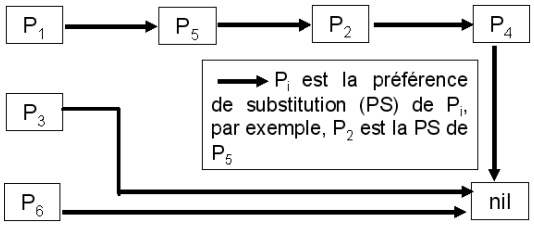


Figure 8.2. Préférences d'un utilisateur et préférences de substitution qui leur sont associées.

	Profil1	Profil2	Profil3	Profil4	Profil5	Profil6	Profil7	Profil8	Profil9	Profil10	Profil11	Profil12	Profil13	Profil14	Profil15	Profil16	Profil17	Profil18	Profil19	Profil20
P1	x	x	x	x																
P2									x	x	x	x								
P3	x		x		x	x			x	x			x	x			x	x		
P4													x	x	x	x				
P5					x	x	x	x												
P6	x	x			x		x		x		x		x		x		x			x

Tableau 6. Les profils générés par l'algorithme de correspondance contextuelle.

Chaque *profil contextuel* correspond à une session donnée et il est généré en fonction du *contexte d'utilisation* de la session. Par exemple, supposons qu'un des *DM* de l'utilisateur ne supporte que du texte, toutes les préférences liées à l'affichage des images ne seront pas satisfaites lors des sessions pour lesquelles l'utilisateur est connecté à travers ce *DM*. Ces préférences n'apparaîtront donc pas dans les *profils contextuels* générés pour ces sessions ; elles appartiennent à la *liste de préférences rejetées*. Le Tableau 6 montre par exemple que le *profil1* est composé des préférences P_1, P_3 et P_6 (P_2, P_4 et P_5 sont dans la *liste de préférences rejetées*) alors que le *profil19* est « vide » (il ne contient aucune préférence).

S'il n'y a pas de *chaînes de préférences de substitution* (aucune préférence n'a de *préférence de substitution*) dans un ensemble de *p* préférences, alors le nombre de profils (*P*) sera calculé de la manière suivante :

$$P = f(p) = \sum_{i=1}^{p} C_p^i = \sum_{i=1}^{p} \binom{p}{i} = \sum_{i=1}^{p} \frac{p!}{(p-i)!\,i!}$$

Pour calculer le nombre de *profils contextuels* que l'algorithme génère à partir d'un ensemble de préférences qui contient des chaînes de préférences de substitution, nous calculons *P* en utilisant la formule ci-dessus. Le nombre de profils contextuels est obtenu en soustrayant à *P* le nombre de combinaisons qui contiennent une préférence avec une ou plusieurs de ses préférences de substitution. Par exemple, supposons que nous ayons 4 préférences P_1, P_2, P_3 et P_4 et que P_2 soit la préférence de substitution de P_1. Le nombre total

de combinaisons possibles est alors de 15 (4 combinaisons de 1 préférence, 6 de 2, 4 de 3 et 1 de 4). Si le nombre total de combinaisons où P_1 et P_2 apparaissent ensemble ((P_1, P_2) (P_1, P_2, P_3) (P_1, P_2, P_4) (P_1, P_2, P_3, P_4)) est 4, alors le nombre total de profils contextuels générés par l'algorithme (pour des sessions différentes) est 11 (c'est-à-dire, 15-4).

Pour l'exemple de la Figure 8.2, l'algorithme peut générer 20 profils contextuels différents (un profil contextuel par session en tenant compte des caractéristiques contextuelles de cette session). Pour un ensemble de 6 préférences sans chaînes de préférences de substitution, nous aurions 63 profils contextuels différents (6 combinaisons de 1 préférence, 15 de 2, 20 de 3, 15 de 4, 6 de 5 et 1 de 6), mais 43 de ces combinaisons contiennent une préférence avec une ou plusieurs de ses préférences de substitution, d'où les 20 profils contextuels générés par l'*algorithme de correspondance contextuelle.*

Dans la section suivante, nous traitons de la résolution des conflits et autres incompatibilités qui peuvent survenir lors de l'ajout de *préférences de l'utilisateur* au *profil contextuel*.

8.3. Résolution de conflits entre préférences de l'utilisateur

On peut reprocher à l'algorithme présenté précédemment sa relative simplicité dans la mesure où il ne met pas en évidence la façon dont sont gérés certains problèmes pouvant survenir lors du traitement des préférences[82]. Dans cette section, nous présentons quelques conflits entre préférences et autres incompatibilités que nous avons identifiés et la manière dont le système devrait réagir s'ils se présentaient. Il est important de noter que ces manières de réagir ne sont pas toujours des solutions aux conflits mais qu'elles permettent aux utilisateurs d'être pour le moins informés de la présence de conflits.

Parmi les causes possibles de conflits, nous mentionnons :

- **L'incompatibilité entre les *préférences de l'utilisateur* et ses droits d'accès au *Système d'Information*.** Par exemple, une préférence inclut une demande d'information dont l'accès est interdit à l'utilisateur. Dans ce cas, les *préférences de l'utilisateur* ne seront pas satisfaites et il n'obtiendra pas l'information souhaitée.
- **L'apparition d'un conflit suite à l'ajout d'une préférence dans le profil contextuel.** Par exemple, une préférence peut consister à obtenir l'information « *seulement* » dans un format *k* alors qu'une autre préférence, dans le profil contextuel, consiste à obtenir l'information « *seulement* » dans un format *j*, avec *k* différent et incompatible avec *j*. En cas de priorités égales pour les préférences en conflit, une solution consiste à choisir la préférence qui peut être satisfaite par rapport aux formats supportés par le dispositif d'accès.
- **L'incompatibilité entre les préférences et les contraintes techniques du *DM***. Par exemple, le format d'affichage souhaité par l'utilisateur n'est pas supporté par son *DM*. Dans ce cas, le système analyse les éventuelles préférences de substitution de la préférence incriminée et vérifie s'il en existe une qui peut être satisfaite. Dans le cas où l'information ne peut pas être affichée dans un

[82] On notera toutefois que l'introduction de priorités entre les préférences (cf. section 8.2.2) constitue déjà une réponse pour la gestion des conflits.

format supporté par le *DM*, un message indique l'impossibilité d'affichage de l'information et la cause d'une telle impossibilité.

– **L'incompatibilité entre les préférences de format exprimées par l'utilisateur et la disponibilité de l'information demandée**. Le système affiche l'information en utilisant le format dans lequel elle est disponible et cherche, parmi les *préférences d'affichage* de l'utilisateur, les caractéristiques qu'il peut appliquer pour ce format.

Généralement, si une préférence génère un conflit comme ceux mentionnés ci-dessus, cette préférence ne sera pas ajoutée au *profil contextuel* mais à la *liste de préférences rejetées*. Le *Système de Gestion de Profils Contextuels* doit connaître les manières de réagir à l'apparition d'un tel conflit afin de le résoudre. En termes généraux, parmi les solutions, nous mentionnons les suivantes :

– Si l'utilisateur n'a pas le droit d'accéder à l'information dans son format préféré (ou si l'information n'est pas disponible dans ce format), le système affiche l'information dans un format autorisé et dans lequel l'information est disponible. Le système consultera les *préférences de l'utilisateur* pour ce format. Par exemple, si le format d'affichage autorisé est du texte, le système affichera l'information en tenant compte de la *préférence de l'utilisateur* définie pour ce format ;
– Le système affichera un message dans le cas où l'information n'est pas supportée par le *DM* ou dans le cas où elle n'est pas disponible dans le système (un message indiquant l'impossibilité d'affichage de l'information et sa cause).

Ainsi, afin de décider si une préférence « *peut être satisfaite* » (*cf.* section 8.2.3, lignes `(7)` et `(19)`), une base de connaissances recense les conflits pré-identifiés lors de la conception. Un *conflit* y est défini à travers un tuple :

```
Conflit (nom, cause, manière de réagir)
```

et exprimé sous la forme d'un template *JESS* de la manière suivante :

(*deftemplate Conflict*
(*slot nom*)
(*slot cause*)
(*slot manière_de_réagir*))

Afin de lever des conflits, l'algorithme de *correspondance contextuel* pourrait d'une part, exploiter cette connaissance et d'autre part, considérer des *priorités* établies entre les *préférences de l'utilisateur*.

Le *Système de Gestion de Profils Contextuels* pourrait utiliser des *priorités* (cf. section 8.2.2) afin de vérifier les préférences : les *préférences spécifiques* ont la priorité la plus haute, suivies des *préférences générales*. Après cette vérification, le *profil contextuel de l'utilisateur* est généré. Le système vérifie :

– D'abord si les activités souhaitées par l'utilisateur (représentées à travers ses préférences d'activité) peuvent être réalisées dans le contexte de la session (notamment en fonction des droits d'accès).
– Puis, si les résultats de ces activités sont disponibles et s'ils peuvent être affichés aux formats indiqués dans les préférences de résultat.
– Enfin, si le dispositif d'accès supporte les caractéristiques des médias exprimés dans les préférences d'affichage pour les formats indiqués.

L'établissement des priorités permet également à l'utilisateur de proposer des alternatives dans le cas où ses préférences ne pourraient pas être supportées par son *DM*. Ces alternatives sont exprimées, par exemple, à travers une *préférence de substitution* dans le cas des *préférences d'affichage*.

8.4. Système de Gestion de Profils Contextuels

Dans cette section, nous présentons les éléments nécessaires à la mise en œuvre d'un *Système de Gestion de Profils Contextuels* (*SGPC*) (cf. Figure 8.3). Ce dernier est un composant (intégrable dans l'architecture d'un *Système d'Information* ou sollicité par le *Système d'Information* en tant que composant externalisé[83]) qui opère en amont d'un composant dédié à la mise en œuvre de l'adaptation dans le *Système d'Information*. Le rôle du *Système de Gestion de Profils Contextuels* est de fournir les éléments permettant de procéder à l'adaptation (à l'utilisateur et à son *contexte d'utilisation*) de l'information gérée au sein du *SI*.

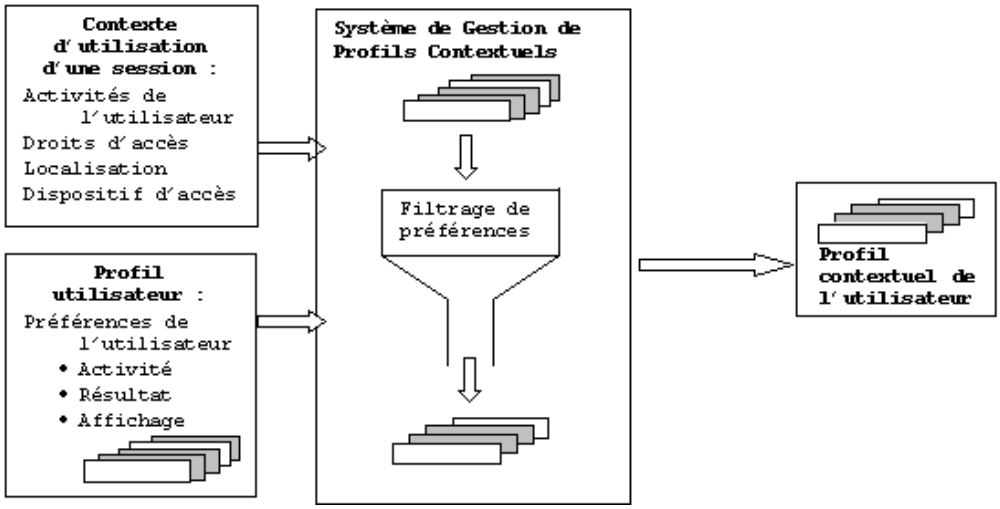


Figure 8.3. Système de Gestion de Profils Contextuels (*SGPC*).

Le *Système de Gestion de Profils Contextuels* prend en entrée un *contexte d'utilisation* et un *profil utilisateur* (ce dernier étant composé de l'ensemble des *préférences de l'utilisateur*). Ensuite, il procède au *filtrage de préférences*, compte tenu du *contexte d'utilisation*, par application d'un algorithme de *correspondance contextuelle*. Seules les *préférences* du *profil utilisateur* qui peuvent être appliquées, étant donné le *contexte d'utilisation*, sont retenues pour constituer le *profil contextuel* de l'utilisateur. Ce *profil contextuel*, artefact produit par le *Système de Gestion de Profils Contextuels*, constitue une représentation réaliste de ce à quoi doit être adaptée l'information, c'est-à-dire *i)* à l'utilisateur évoluant au moment de sa demande dans un contexte particulier et *ii)* à ce contexte lui-même. Le caractère réaliste de cette représentation tient justement au fait que ce *profil* est *contextuel*, c'est-à-dire reconstruit pour chaque session de l'utilisateur sur la base des *préférences* les plus pertinentes par rapport au *contexte d'utilisation* courant. Si la notion de *profil utilisateur* est largement exploitée comme donnée d'entrée d'un processus d'adaptation, nous voyons la contextualisation de ce *profil utilisateur* comme un moyen d'affiner ce processus, de le

[83] L'externalisation du *SGPC* peut ainsi permettre le partage de profils entre applications.

rendre dynamique et évolutif : en effet, l'adaptation suit les évolutions du contexte (un changement de *DM* d'accès par exemple).

Du point de vue de l'adaptation à l'utilisateur, nous nous concentrons uniquement dans ce travail sur un *profil utilisateur* constitué de *préférences*, c'est-à-dire d'expressions traduisant les souhaits de l'utilisateur à différents égards. Nous définissons ainsi des *préférences* portant sur les activités de l'utilisateur, ou encore sur la nature et la forme du contenu délivré. En nous focalisant sur les préférences exprimées par l'utilisateur, nous ne considérons pas ici un processus d'adaptation reposant, par exemple, sur des caractéristiques personnelles telles que des niveaux de compétences, l'âge, le sexe, *etc*. A l'image des modes d'acquisition de profil présentés dans Kassab *et al.* [Kass05], le *Système de Gestion de Profils Contextuels* est conçu de telle sorte que cette information peut être :

- Fournie par l'utilisateur à travers des interfaces dédiées ;
- Définie au moyen de profils généraux d'utilisateurs ;
- Déduite de l'historique de l'utilisateur, c'est-à-dire de ses sessions précédentes [Bouc06] [Tami06].

Plusieurs travaux tiennent compte du *contexte d'utilisation* afin de personnaliser l'information [Bouc06] [Tami06] [Kass05] [Bouz05] [Kech06]. Dans notre approche, nous utilisons les caractéristiques du *contexte d'utilisation*, constitué essentiellement de quatre types d'informations : les descriptions des activités dont l'utilisateur demande l'exécution au cours d'une session, les droits de l'utilisateur sur les données du ou des *Systèmes d'Information* collaborant avec le *Système de Gestion de Profils Contextuels*, la localisation de l'utilisateur (coordonnées *GPS* par exemple) et les caractéristiques du dispositif utilisé par l'utilisateur au cours de la session (inspirées des extensions *CC/PP* fournies par Indulska *et al.* [Indu03a]). Dans notre proposition, nous ne présentons pas de façon détaillée le modèle de *contexte d'utilisation,* issu des travaux de Kirsch-Pinheiro [Kirs06], dans la mesure où une description à un niveau global de ce modèle suffit à la compréhension de l'approche que nous défendons.

Dans la section suivante, nous illustrons la représentation des *préférences d'un utilisateur* médecin et la génération de son profil contextuel lors de la consultation des analyses médicales d'un de ses patients.

8.5. Exemple

Considérons le scénario suivant : un médecin généraliste (appelé *docteur Thierry Dupont*) souhaite consulter une « *échographie* » prescrite à un patient.

Les *préférences d'activité* et *de résultat* du docteur Dupont sont définies comme suit (cf. section 8.1) :

P_1= Le docteur Dupont souhaite obtenir les images et la vidéo de l'échographie.

```
P1= Préférence_Résultat (spécifique, "Consulter_échographie",
<(résult_écho, PAff_image),(résult_écho, PAff_vidéo)>)
```

P_2 = Le docteur Dupont souhaite obtenir le texte de l'analyse préparé par le radiologiste qui a pratiqué cette échographie.

```
F1= Consulter_analyse_échographie (< nom_patient, échographie, date>,
< analyse du radiologiste >)
P2= Préférence_Activité (spécifique, (), F1)
```

P_3 = Le docteur Dupont souhaite obtenir l'analyse des échographies précédentes dans un format graphique (images ou vidéo).

```
P3=Préférence_Résultat (spécifique,  "Consulter_échographie_précédente",
< (résult_écho, PAff_image), (résult_écho, PAff_vidéo)>)
```

P_4 = Lorsque le docteur Dupont demande les résultats d'une échographie, il souhaite aussi consulter le dernier enregistrement (dans le dossier médical du patient) du médecin qui a prescrit cette échographie.

```
F2 = Consulter_échographie (<nom_patient, date>, < résult_écho >)
F3 = Consulter_dernier_enregistrement (<nom_patient, médecin,échographie> ,
< dernier_enregistrement >)
P4 = Préférence_Activité (générale, (), F2;F3)
```

P_5 = Lorsque le docteur Dupont demande les résultats d'une échographie, il souhaite aussi consulter les échographies précédentes du même type (par exemple, si l'échographie est rénale, les résultats des échographies rénales précédentes sont à présenter).

```
F4 = Consulter_écho_précédente (<nom_patient, type >, < résult_écho >)
P5 = Préférence_Activité (générale, (), F2;F4)
```

Nous supposons que les *préférences d'affichage* du docteur Dupont sont : {P_{1Af}, P_{2Af}, P_{3Af}, P_{4Af}} où :

P_{1Af} exprime les caractéristiques des images que le *Docteur Dupont* veut pour la session en cours.

```
P1Af = Préférence_Affichage (spécifique, image, {JPEG, 100, 200}, P2Af)
```

P_{2Af} exprime que, si pour la session en cours son *DM* ne supporte pas les images, alors l'information sera affichée en format texte. C'est-à-dire, P_{2Af} est une *préférence de substitution* de P_{1Af} et montre les caractéristiques du texte qui sera affiché dans le cas où le *DM* ne supporte pas les images.

```
P2Af = Préférence_Affichage (spécifique, texte, {Arial, 10, bleu, .doc}, nil)
```

P_{3Af} exprime les préférences du docteur Dupont par rapport à l'affichage du texte, pour la session en cours.

```
P3Af = Préférence_Affichage (spécifique, texte, {Times, 8, noir, .txt}, nil)
```

Finalement, P_{4Af} exprime ses préférences par rapport à l'affichage du texte, pour toutes ses sessions.

```
P4Af = Préférence_Affichage (générale, texte, {Tahoma, 7, vert, .doc}, nil)
```

Le système analyse chaque préférence, de P_{1Af} à P_{4Af}. P_{1Af}, P_{2Af} et P_{3Af} ont une priorité supérieure à P_{4Af} puisque les premières sont des préférences *spécifiques* et la dernière est une préférence *générale*. Si les images spécifiées en P_{1Af} sont supportées par le *DM*, cette préférence est ajoutée au *profil contextuel* et P_{2Af} est ajoutée à la *liste de préférences rejetées*. Sinon, P_{1Af} est ajoutée à la *liste de préférences rejetées* et P_{2Af} est incluse dans le *profil contextuel*. De plus, pour la session en cours, le texte sera affiché selon les caractéristiques définies par P_{3Af} (puisque P_{3Af} étant *spécifique* à la session en cours, elle a une priorité supérieure à celle de P_{4Af}).

Les *préférences d'activité* et *de résultat* de l'utilisateur sont contraintes par les droits d'accès de l'utilisateur dans le système. Aussi, le système doit vérifier si également l'utilisateur peut ou non accéder à l'information. Certains droits d'accès concernent les règles établies par l'établissement à laquelle l'utilisateur ou le système appartient. Dans cet exemple, l'hôpital a établi certaines règles concernant la consultation des

analyses médicales par les différentes catégories de médecin. Une analyse médicale est définie à travers le tuple :

```
Analyse_Médicale (type, prescrit_par)
```

et exprimée sous la forme d'un template *JESS* de la manière suivante :

(*deftemplate Analyse_Médicale*
(*slot type*)
(*slot prescript_par*))

où *type* peut prendre pour valeurs « *échographie* », « *analyse de sang* », *etc*. et *prescrit_par* est le médecin qui a prescrite l'analyse. Un médecin est représenté par le tuple :

```
Médecin (nom, type)
```

et exprimé sous la forme d'un template *JESS* de la manière suivante :

(*deftemplate Médecin*
(*slot nom*)
(*slot categorie*))

où *nom* correspond au médecin et *catégorie* prend pour valeur « *généraliste* » ou « *spécialiste* ». Les règles pour les médecins qui accèdent aux *SIW* sont exprimées de la manière suivante[84] :

```
Si (Analyse_Médicale.prescrit_par.nom == Médecin.nom) ou
   (Médecin.categorie == "spécialiste") ou
   (Analyse_Médicale.prescrit_par.categorie == Médecin.categorie)
      alors accès_permis
Si (Analyse_Médicale.prescrit_par.categorie == "spécialiste" et
    Médecin.categorie == "généraliste")
      alors accès_refusé
```

Les règles sont décrites en *JESS* de la manière suivante :

Pour permettre l'accès au médecin :

(defrule décider_accès_permis
(*Analyse_Médicale* (*type ?tam*) (*prescript_par ?pp*)) //*Connaître l'analyse médicale*
(*Médecin* (*nom ?nomm*) (*categorie ?tm*)) *// Connaître le premier médecin*
(*Médecin* (*nom ?nomm2*) (*categorie ?tm2*)) *// Connaître le deuxième médecin*
(*Demande* (*nom_médecin ?inm*) (*analyse_demandé ?ad*)) *//Médecin « inm » qui demande l'analyse médicale « ad »*
(or
(and (test (eq ?ad ?tam)) (test (neq ?pp ?inm)) (test (eq ?pp ?nomm2)) (test (eq ?inm ?nomm))
(test (eq ?tm ?tm2)) (test (neq ?nomm ?nomm2))) // Si le médecin demandant l'analyse médicale est du même type que le médecin qui l'a prescrite.
(and (test (eq ?nomm ?inm)) (test (eq ?tm "spécialiste"))) // Si le médecin demandant l'analyse médicale est de type « spécialiste »
(and (test (eq ?ad ?tam)) (test (eq ?pp ?inm)) (test (eq ?nomm ?inm))) // Si le médecin demandant l'analyse médicale est le même que celui qui l'a prescrite.
)=> (assert (acces (nom_médecin ?inm) (type "permis"))))

[84] L'expression des règles et des faits telle qu'elle apparaît ici est définie dans la section 6.2.4.2 et [Carr06].

Afin de montrer de manière plus simple la règle qui permet à l'utilisateur l'accès au système, nous l'avons décomposée en trois règles comme suit :

Première sous-règle. Si le médecin demandant l'analyse médicale est du même type que le médecin qui l'a prescrite :

```
(defrule decider_acces_permis_c1
( Analyse_Médicale (type ?tam) (prescript_par ?pp)) //Connaître l'analyse médicale
( Médecin (nom ?nomm) (categorie ?tm)) // Connaître le premier médecin
( Médecin (nom ?nomm2) (categorie ?tm2)) // Connaître le deuxième médecin
( Demande (nom_médecin ?inm) (analyse_demandé ?ad)) //médecin « inm » qui demande
                                                    l'analyse médicale « ad »
(and
(test (eq ?ad ?tam)) // la même analyse médicale
(test (neq ?pp ?inm)) // le médecin qui demande l'analyse  n'est pas le même que celui qui l'a prescrite
(test (eq ?pp ?nomm2)) // le médecin qui l'a prescrite existe dans le système
(test (eq ?inm ?nomm)) // le médecin qui la demande existe dans le système
(test (neq ?nomm ?nomm2)) // leurs noms sont différents
(test (eq ?tm ?tm2)) // le type de médecin est le même
) => (assert (acces (nom_médecin ?inm) (type "permis"))))
```

Deuxième sous-règle. Si le médecin qui demande l'analyse médicale est spécialiste :

```
(defrule decider_acces_permis_c2
( Analyse_Médicale (type ?tam) (prescript_par ?pp)) //Connaître l'analyse médicale
( Médecin (nom ?nomm) (categorie ?tm)) // Connaître le premier médecin
( Demande (nom_médecin ?inm) (analyse_demandé ?ad)) //Médecin « inm » qui demande
                                                    l'analyse médicale « ad »
(and
(test (eq ?nomm ?inm))  // Le médecin qui la demande existe dans le système
(test (eq ?tm "spécialiste")))  // et il est de type « spécialiste »
=> (assert (acces (nom_médecin ?inm) (type "permis"))))
```

Troisième sous-règle. Si le médecin qui la demande est le même qui l'a prescrite :

```
(defrule decider_acces_permis_c3
( Analyse_Médicale (type ?tam) (prescript_par ?pp)) //Connaître l'analyse médicale
( Médecin (nom ?nomm) (categorie ?tm)) // Connaître le premier médecin
( Demande (nom_médecin ?inm) (analyse_demandé ?ad)) //Médecin « inm » qui demande
                                                    l'analyse médicale « ad »
(and
(test (eq ?ad ?tam)) // La même analyse médicale
(test (eq ?pp ?inm)) // Le médecin qui la demande est le même qui l'a prescrite
(test (eq ?nomm ?inm))) // Le médecin qui la demande existe dans le système
=> (assert (acces (nom_médecin ?inm) (type "permis"))))
```

Nous venons de décrire les trois sous-règles qui permettent à l'utilisateur l'accès au système. La sous-règle suivante exprime l'interdiction d'accès au système pour le médecin :

```
(defrule decider_acces_refusé
(Analyse_Médicale (type ?tam) (prescript_par ?pp)) //Connaître l'analyse médicale
(Médecin (nom ?nomm) (categorie ?tm)) // Connaître le premier médecin
(Médecin (nom ?nomm2) (categorie ?tm2)) // Connaître le deuxième médecin
(Demande (nom_médecin ?inm) (analyse_demandé ?ad)) //Médecin « inm » qui demande
```

l'analyse médicale « ad »

```
(and
(test (eq ?ad ?tam)) // Test de comparaison de deux analyses médicales
(test (neq ?pp ?inm)) // Le médecin qui la demande n'est pas le même qui l'a prescrite
(test (eq ?pp ?nomm2)) // Le médecin qui l'a prescrite existe dans le système
(test (eq ?inm ?nomm)) // Le médecin qui la demande existe dans le système
(test (neq ?nomm ?nomm2)) // Les noms des deux médecins sont différents
(test (eq ?tm2 "spécialiste")) //Le médecin qui l'a prescrite est de type « spécialiste »
(test (eq ?tm "généraliste"))) // Le médecin qui la demande est de type « généraliste »
=> (assert (acces (nom_médecin ?inm) (type "refusé"))))
```

Concernant le *DM* du Docteur Dupont, nous supposons deux faits. Premièrement, son *PDA* (DM_1) supporte les formats vidéo et image, mais ne peut pas les afficher en même temps. Deuxièmement, son téléphone portable (DM_2) ne supporte que du texte. Ces faits (cf. note de bas de page numéro 82) sont représentés comme suit :

```
(assert ( Formats_supportés
(DM "DM1")
(formats "vidéo" "texte" "son" "images")))

(assert ( Formats_non_supportés
(DM "DM1")
(formats "vidéo&images")))

(assert ( Formats_supportés
(DM "DM2")
(formats "texte")))

(assert ( Formats_non_supportés
(DM "DM2")
(formats "vidéo" "son" "images")))
```

Ces faits se basent sur les templates *JESS*, définis de la manière suivante :

```
(deftemplate Formats_supportés
(slot DM)
(multislot formats))

(deftemplate Formats_non_supportés
(slot DM)
(multislot formats))
```

Afin de définir le *profil contextuel* du docteur Dupont, le système doit analyser ses préférences, et notamment, les comparer aux formats supportés par son *DM*. Pour une session « s_1 », le profil contextuel PC_1 ("*Docteur Thierry Dupont*", s_1, DM_1) est composé de {P_1, P_2, P_3, P_4 et P_5} (définies précédemment) après l'analyse réalisée par le système :

– P_1 et P_2 sont satisfaites puisque DM_1 supporte la vidéo, le texte et les images.

- P_3 peut aussi être satisfaite dès lors qu'on suppose, pour l'exemple, que les échographies précédentes ont été prescrites par le docteur Dupont ou par d'autres médecins généralistes (selon les règles de l'hôpital)[85].
- De façon similaire, P_4 peut être aussi satisfaite sous réserve que le médecin qui a prescrit cette échographie et qui a fait le dernier enregistrement est un médecin généraliste.
- P_5 est aussi satisfaite si les échographies précédentes du même type ont été prescrites par lui ou par d'autres médecins généralistes.

Supposons à présent que les *préférences de résultat* P_6 et P_7 sont respectivement associées à F_3 dans P_4 et à F_4 dans P_5. On définit :

```
P6=Préférence_Résultat(spécifique,"Consulter_dernier_enregistrement",
<(dernier_enregistrement,nil)>)
P7= Préférence_Résultat (spécifique, "Consulter_écho_précédente",
<( résult_écho,nil)>)
```

Ni P_6 ni P_7 n'ont été associées à des *préférences d'affichage* (l'utilisateur n'en a pas définies). Le système change d'abord la valeur de « `<( résult_écho,nil)>)` » par « `<(résult_écho, PAff_image),(résult_écho, PAff_vidéo)>)` » en P_7, puisque le docteur Dupont demande en P_1 que toute information liée à une échographie soit affichée sous forme d'images ou de vidéo (`PAff_image` correspond donc en fait à P_{1Af}). Ensuite, le système change la valeur « *nil* » en P_6 par P_{3Af} car le dernier enregistrement du dossier médical du patient est récupéré en format texte.

Pour le second exemple de génération de profil, nous supposons à présent que le docteur Dupont est connecté à travers son téléphone portable (DM_2), qui ne supporte que du texte. Le profil contextuel PC_2 ("*Docteur Thierry Dupont*", s_2, DM_2) est composé des *préférences d'activité* et *de résultat* {P_2, P_3, P_4 et P_5} pour les raisons suivantes :

- P_1 ne peut pas être satisfaite car DM_2 ne supporte que du texte.
- P_2 peut être satisfaite et le docteur Dupont obtiendra le texte de l'analyse faite par le radiologiste.
- P_3 peut être satisfaite si les échographies précédentes ont été prescrites par le docteur Dupont ou par d'autres médecins généralistes[86] mais les résultats des échographies précédentes seront affichés en texte.
- P_4 peut être satisfaite. Le docteur Dupont peut consulter le dossier médical du patient, et plus particulièrement, le dernier enregistrement du médecin qui a prescrit cette échographie.
- P_5 peut être satisfaite. Le docteur Dupont peut accéder aux résultats des échographies précédentes du même type, prescrites par lui ou d'autres généralistes, mais seulement en format texte.

Le système analyse aussi les *préférences d'affichage* : P_{1Af} ne sera pas satisfaite car DM_2 ne supporte que du texte. P_{2Af} et P_{3Af} ont une priorité supérieure à P_{4Af} puisque les premières sont des préférences *spécifiques* alors que la dernière est une préférence *générale*. P_{1Af} est ajoutée à la *liste de préférences rejetées* et P_{2Af} est

[85] Si les échographies précédentes ont été prescrites par des médecins spécialistes, alors P_3 n'est pas incluse dans le profil contextuel en raison des règles établies par l'hôpital. Il va de même pour les cas de P_4 et de P_5.

[86] Avec une hypothèse similaire à celle formulée pour le Profil Contextuel PC_1.

incluse dans le *profil contextuel*. De plus, pour la session en cours, le texte qui remplace l'information demandée sous format vidéo sera affichée compte tenu des caractéristiques définies en P_{2Af} (puisque P_{2Af} est la préférence de substitution de P_{1Af}, spécifique à la session en cours et a donc une priorité supérieure à celle de P_{4Af}). P_{4Af} est ajoutée à la *liste de préférences rejetées* (car elle est une préférence générale et il existe d'autres préférences spécifiques qui décrivent les caractéristiques du texte). P_{3Af} est incluse dans le *profil contextuel*. Le texte qui ne remplace pas d'images sera affiché en utilisant les caractéristiques définies en P_{3Af}.

Finalement, le système change en P_6 (la *préférence de résultat* associée à F_3 dans P_4) la valeur « *nil* » par P_{3Af} car le dernier enregistrement du dossier médical du patient est récupéré sous format texte, et en P_7 (la *préférence de résultat* associé à F_4 dans P_5) par P_{2Af} car DM_2 ne supporte que du texte.

8.6. Intégration de l'adaptation à PUMAS

Dans cette section, nous montrons comment sont intégrées les fonctionnalités du *Système de Gestion de Profils Contextuels* aux différents composants du *framework PUMAS*. Le *Système de Gestion de Profils Contextuels* est une fonctionnalité du *SMA d'adaptation* de *PUMAS*.

Les capacités d'adaptation reposent sur un processus en deux étapes. Premièrement, le *filtre de contenu* permet de sélectionner l'information pertinente compte tenu du profil de l'utilisateur défini dans le système. Ce filtre se base sur la sélection du contenu le plus approprié par rapport aux préférences d'*activité* et de *résultat* de l'utilisateur. Deuxièmement, le *filtre d'affichage*, qui s'applique aux résultats du premier filtre, considère les caractéristiques et contraintes techniques du dispositif d'accès. Ce filtre concerne l'affichage correct de l'information en prenant en compte les préférences d'*affichage* de l'utilisateur qui sont satisfaites en fonction des caractéristiques du dispositif d'accès. Ces filtres sont appliqués par les agents de *PUMAS* comme présenté dans la Figure 8.4 :

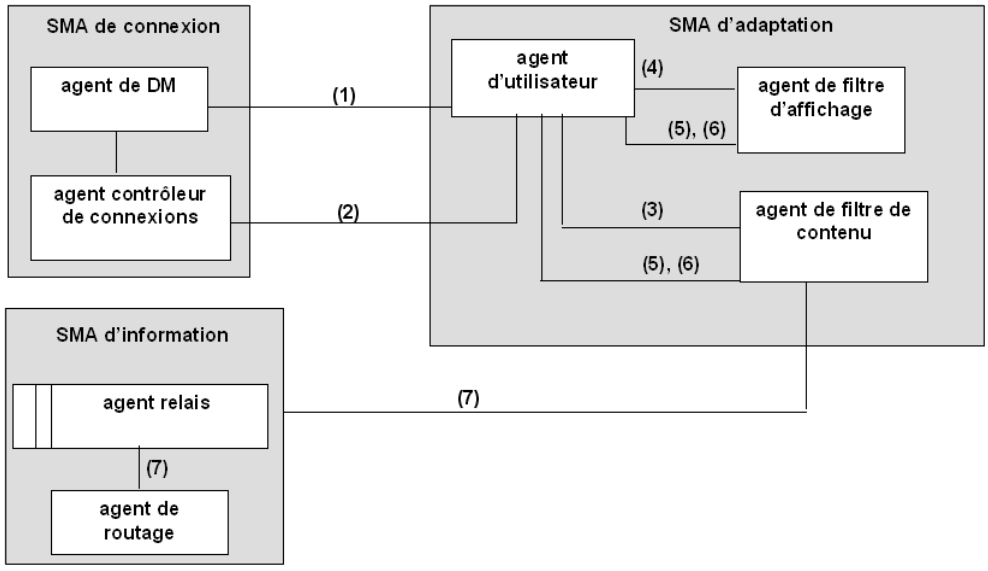


Figure 8.4. Processus de filtrage d'information à travers l'analyse des préférences de l'utilisateur.

(1) L'*agent de DM* envoie le fichier *d'utilisateur* (contenant les *préférences de l'utilisateur*) et le fichier *de session* (contenant les caractéristiques de la session) à l'*agent d'utilisateur*.

(2) S'il n'y a pas de fichier *de session*, l'*agent d'utilisateur* demande cette information à l'a*gent contrôleur de connexions* (contrôlant toutes les connexions des dispositifs).

(3) L'*agent d'utilisateur* vérifie s'il y a des conflits entre les *préférences d'activité* et *de résultats* à l'aide de l'*agent de filtre de contenu*. Ce dernier confronte chaque *préférence d'activité* et *de résultat* aux droits d'accès de l'utilisateur dans le système.

(4) L'*agent d'utilisateur* vérifie s'il y a des conflits entre les *préférences d'affichage* à l'aide de l'*agent de filtre d'affichage*. Ce dernier évalue chaque *préférence d'affichage* en tenant compte des caractéristiques du dispositif d'accès (définies dans le fichier *de dispositif*) ainsi que de la connaissance (stockée dans sa *base de connaissances*) des problèmes survenus lors des connexions précédentes en utilisant le même type de dispositif.

(5) L'*agent d'utilisateur* vérifie s'il y a des conflits entre les *préférences d'activité* et *de résultats*, et les *préférences d'affichage*. Cette vérification est accomplie à l'aide de l'*agent de filtre de contenu* et de l'*agent de filtre d'affichage*. S'il y a des problèmes liés au format supporté par le *DM*, l'*agent d'utilisateur* change les *préférences de résultat* (la propriété correspondant à la *préférence d'affichage* associée) à l'aide de l'*agent de filtre d'affichage*. L'information sur les formats supportés est extraite du fichier *de dispositif* et la *base de connaissances* de l'*agent de filtre d'affichage*. Lorsque la propriété correspondant à la *préférence d'affichage* associée à la *préférence de résultat* est changée pour celle d'un format supporté par le *DM*, l'*agent de filtre d'affichage* cherche s'il y a des *préférences d'affichage* liées au nouveau format afin de connaître les caractéristiques préférées par l'utilisateur pour ce nouveau format.

(6) L'*agent d'utilisateur*, à l'aide de l'*agent de filtre de contenu* et de l'*agent de filtre d'affichage*, est chargé de : *i)* sélectionner les *préférences d'activité* et *de résultats* pour les envoyer à l'*agent de filtre de contenu, ii)* sélectionner les *préférences d'affichage* pour les envoyer à l'*agent de filtre d'affichage* et finalement, *iii)* générer le profil contextuel de l'utilisateur (en suivant l'*algorithme de correspondance contextuel*, cf. section 8.2.3) sans conflits et en accord avec les *préférences de l'utilisateur* pour l'envoyer à l'*agent de filtre de contenu*. Ce dernier reçoit et sauvegarde le profil contextuel de l'utilisateur dans un historique pour chaque utilisateur. Cet historique est utile pour récupérer (à la place de créer) un profil existant d'un utilisateur si ses préférences et son contexte d'utilisation sont semblables pour la session courante.

(7) Lorsque l'*agent relais* reçoit une requête de l'utilisateur, il demande le profil contextuel de l'utilisateur à l'*agent de filtre de contenu* et enrichit la requête avec l'information sur les préférences appartenant à ce profil. L'*agent relais* ajoute à la requête l'information sur les préférences de l'utilisateur (*activité, résultats* et *affichage*) et envoie cette requête augmentée (c'est-à-dire, « *requête originale* » + « *information sur le préférences l'utilisateur* », cf. section 8.2.1) à l'*agent de routage* qui est en charge d'analyser la requête et de la rediriger aux systèmes d'information capables d'y répondre (cf. section 7, processus de *routage de requêtes*).

8.7. Conclusion

Les concepteurs de *Systèmes d'Information* doivent disposer de mécanismes permettant de représenter l'information sur les souhaits de l'utilisateur par rapport aux activités qu'il veut accomplir dans le système, aux résultats obtenus de ces activités et au format dans lequel ces résultats seront affichés sur les dispositifs mobiles. L'accomplissement de ces souhaits peut être affecté à cause des caractéristiques contextuelles de la session en cours. Les souhaits de l'utilisateur (que nous appelons « *préférences de l'utilisateur* ») sont les composants de base du *profil de l'utilisateur*. Afin de générer un *profil contextuel* de l'utilisateur, les concepteurs doivent disposer de mécanismes qui évaluent chaque *préférence de l'utilisateur* et décident si celle-ci peut être appliquée en fonction des caractéristiques contextuelles. Lors de la création du profil, des conflits entre préférences peuvent survenir (tels que des préférences différentes pour la même information, des incompatibilités entre les préférences et les droits d'accès au système ou encore, des contraintes techniques du dispositif d'accès non compatibles). Pour ces raisons, des mécanismes de gestion de conflits doivent être également implémentés lors de la génération du profil.

Dans ce chapitre, nous avons tout d'abord présenté notre proposition de représentation et classification des *préférences de l'utilisateur* en distinguant les préférences d'*activité* (exprimant la manière dont l'utilisateur veut accomplir ses activités), les préférences de *résultat* (concernant la sélection des résultats de ces activités et les formats souhaités pour chaque résultat) et celles d'*affichage* (correspondant aux caractéristiques du format dans lequel les résultats doivent être affichés sur le dispositif d'accès).

Puis, nous avons montré la manière dont le *profil contextuel* d'un utilisateur est généré à l'aide de l'algorithme de *correspondance contextuelle*, en considérant comme éléments de base les *préférences de l'utilisateur* ainsi que le *contexte d'utilisation* de la session en cours.

Nous avons discuté sur la résolution des conflits qui peuvent apparaître entre les *préférences de l'utilisateur* constitutives du *profil contextuel* obtenu. Parmi ces conflits, nous avons identifié : *i)* l'incompatibilité entre les *préférences de l'utilisateur* et ses droits d'accès au *Système d'Information* ; *ii)* l'apparition d'un conflit suite à l'ajout d'une préférence dans le profil contextuel ; *iii)* l'incompatibilité entre les préférences et les contraintes techniques du *DM* ; *iv)* l'incompatibilité entre les préférences de format exprimées par l'utilisateur et la disponibilité de l'information demandée (lorsque l'information n'est pas disponible dans le format demandé). Nous avons proposé des solutions pour traiter ces conflits. Par exemple, dans le cas où l'information est disponible mais dans un format différent de celui souhaité par l'utilisateur, le système affiche l'information dans le format donné, à condition que le format soit autorisé. Un autre cas de solution consiste à informer l'utilisateur sur la présence d'un conflit (le système montre un message indiquant l'impossibilité d'affichage de l'information et sa cause).

Nous avons défini le *Système de Gestion de Profils Contextuels* (*SGPC*) qui fournit les éléments permettant de procéder à l'adaptation de l'information, compte tenu des *préférences de l'utilisateur* et du *contexte d'utilisation*. Nous avons ensuite illustré les processus de représentation des *préférences de l'utilisateur* et de génération du *profil contextuel* à travers les activités d'un médecin consultant des analyses médicales d'un patient.

Enfin, nous avons montré l'intégration des processus de gestion de préférences et de génération de profils contextuels dans *PUMAS*. Le *SGPC* est une fonctionnalité du *SMA d'adaptation* de *PUMAS*.

Le chapitre suivant présente des aspects pour la modélisation et l'implémentation de *PUMAS* et du *Système de Gestion de Profils Contextuels*. Nous montrons nos choix en termes d'outils d'implémentation et quelques interfaces pour l'utilisateur de *PUMAS* lui permettant de se connecter, d'envoyer des messages à d'autres utilisateurs et de fournir ses préférences pour une session.

9. MISE EN ŒUVRE DE PUMAS ET DE LA GESTION DES PREFERENCES DE L'UTILISATEUR

Ce chapitre présente les aspects les plus importants de modélisation et d'implémentation du *framework PUMAS* ainsi que du *Système de Gestion de Profils Contextuels* (*SGPC*). Nous justifions les utilisations, d'une part, de *Protégé*, l'outil que nous avons choisi pour la gestion des connaissances et, d'autre part, de *JADE-LEAP*, la plate-forme permettant l'exécution d'agents. Nous décrivons brièvement les plates-formes des *Systèmes Multi-Agents* (*SMA*) qui suivent les spécifications *FIPA* en soulignant la raison pour laquelle nous avons choisi *JADE-LEAP* pour l'implémentation de *PUMAS*. Enfin, nous présentons des interfaces pour l'accès à la plate-forme à travers des *DM* et pour la gestion des préférences de l'utilisateur.

9.1. PUMAS – Implémentation

Dans cette section, nous présentons tout d'abord les diagrammes *AUML* de *PUMAS*.

9.1.1 Modélisation des Systèmes Multi-Agents

Nous utilisons ici *Agent UML* (*AUML*) [Odel01] [Pogg04] comme formalisme pour modéliser l'interaction entre les agents et pour expliquer la manière dont les agents *PUMAS* peuvent accéder à un *SIWA* afin d'obtenir l'information la plus pertinente. La Figure 9.1 montre le diagramme de *packages AUML*[87] (avec leurs classes et leurs relations) qui représente *PUMAS*.

[87] Un diagramme de packages plus détaillé est disponible à l'adresse suivante : *http://www-lsr.imag.fr/users/Angela.Carrillo/*

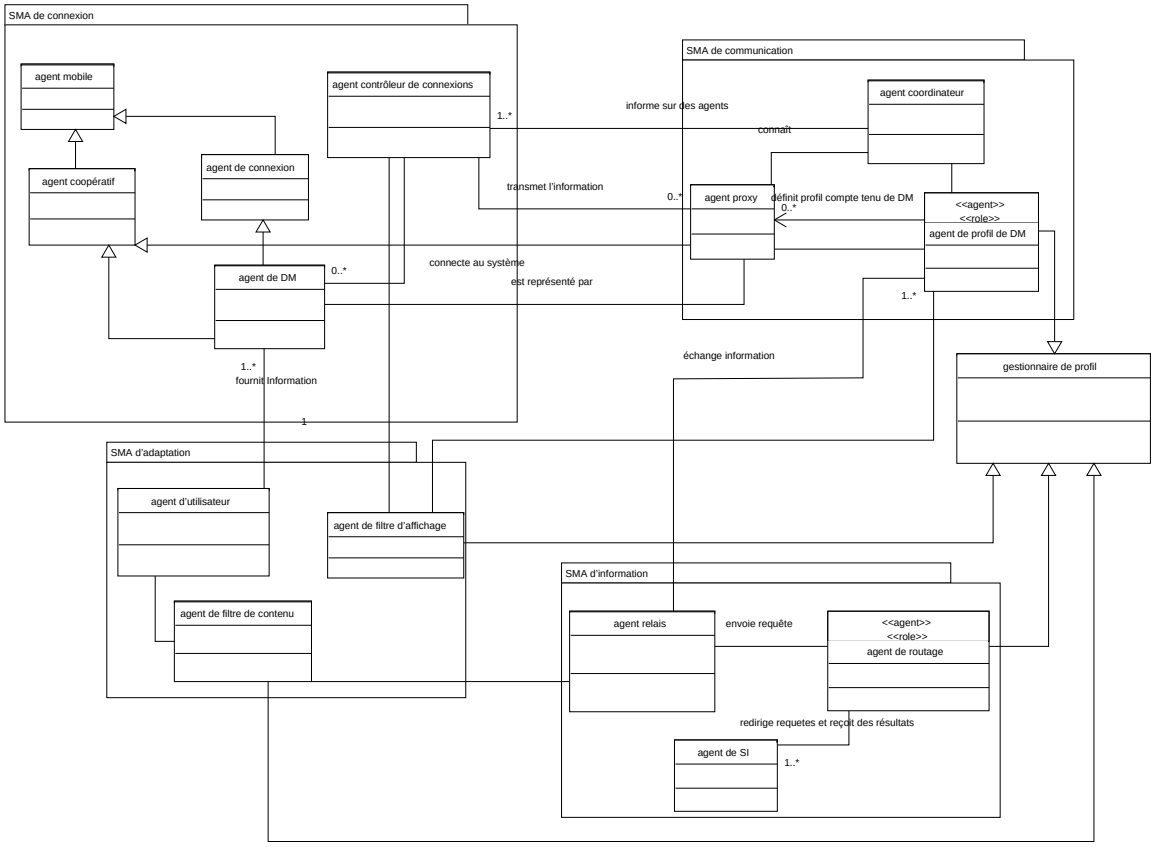


Figure 9.1. Diagramme de packages de PUMAS.

Les Figures suivantes (de 9.2 à 9.5) présentent en détail les agents (représentés à la manière de classes) de chaque *SMA* (représenté à la manière de *packages*) avec leurs propriétés et leurs méthodes :

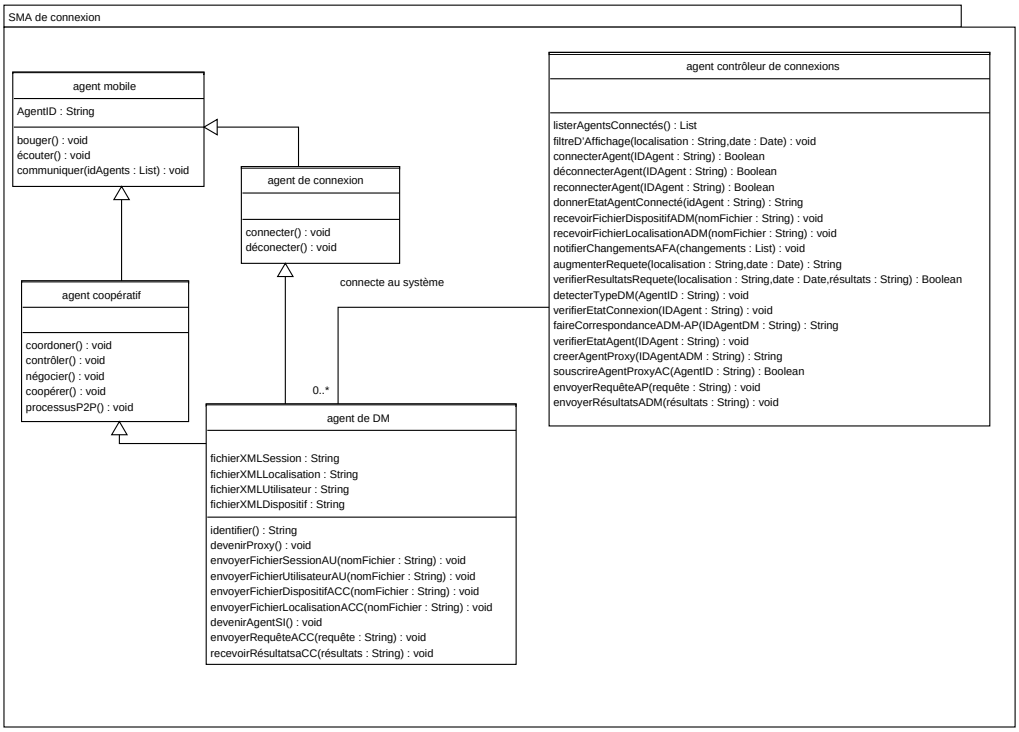


Figure 9.2. SMA de connexion de PUMAS.

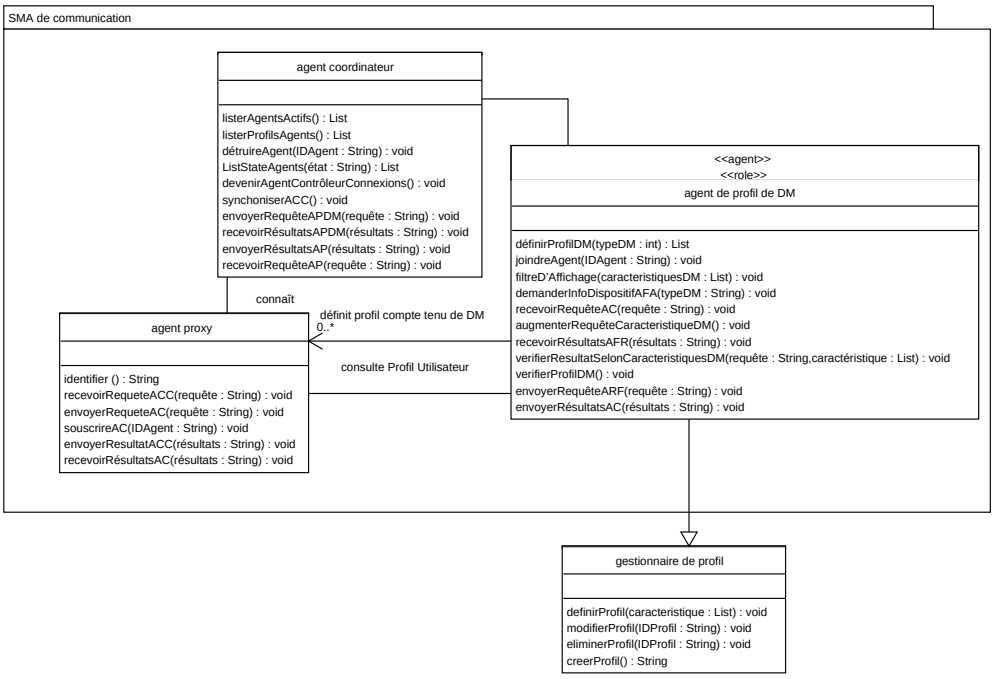


Figure 9.3. SMA de communication de PUMAS.

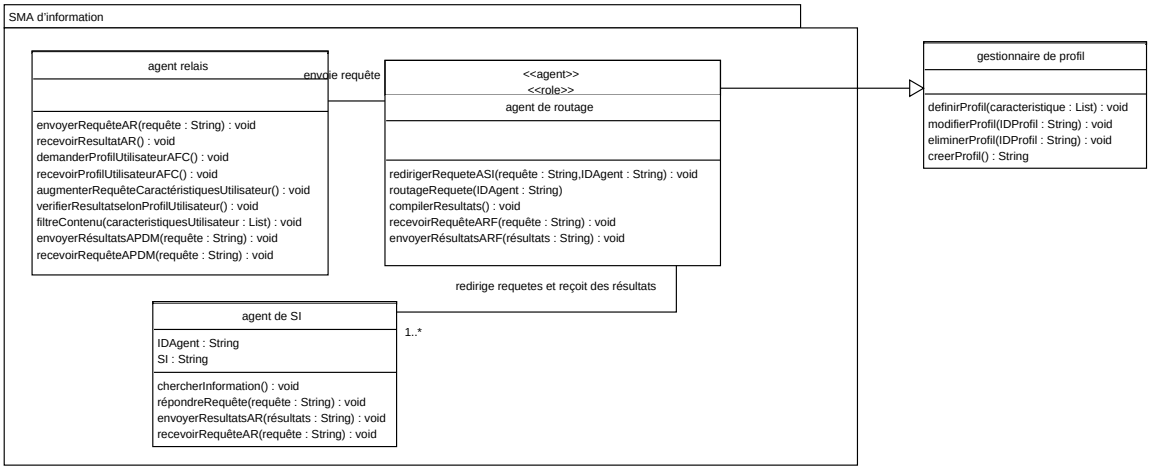


Figure 9.4. SMA d'information de PUMAS.

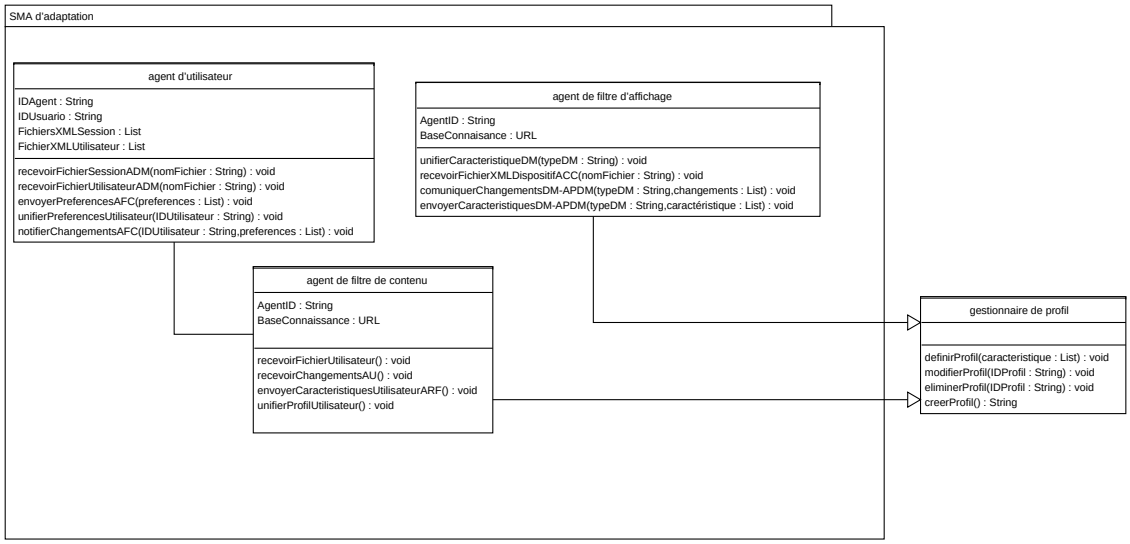


Figure 9.5. SMA d'adaptation de PUMAS.

Dans les diagrammes d'interaction d'*AUML*, les messages entre les agents sont nommés *actes de communication* (par exemple, « *confirm* », « *disconfirm* », « *inform* », « *not-understand* », « *propose* », « *refuse* », « *request* », « *subscribe* », « *propagate* », cf. section 3.3). Dans ce type de diagramme, des messages peuvent impliquer une condition et/ou représenter la concurrence entre les exécutions des agents. Cette concurrence est symbolisée par plusieurs lignes de vie associées à la classe de l'agent. Par exemple, la Figure 9.6 montre la concurrence d'exécution de plusieurs agents de *DM*. Nous expliquons cela à travers l'exemple suivant (cf. Figure 9.6) : plusieurs *agents de DM* pourraient simultanément envoyer leurs messages de requête à l'*agent contrôleur de connexions*. La Figure 9.6 montre trois *agents de DM* envoyant chacun un message « *request* » à l'*agent contrôleur de connexions*. La boîte de décision (symbolisée par un losange) reflète qu'un *agent de DM* peut choisir le message à envoyer selon

une ou plusieurs conditions (la Figure 9.6 ne montre que les possibles messages que l'*agent de DM* envoie à l'*agent contrôleur de connexions*, les conditions peuvent être exprimées à travers des commentaires). Dans la Figure 9.6, un *agent de DM* peut ainsi envoyer un message « *propose* » (par exemple, proposer d'être représenté par le même *agent proxy* à chaque connexion), ou un message « *subscribe* » (par exemple, s'inscrire comme un utilisateur valide dans le système) ou un message « *propagate* » (par exemple, propager une requête formulée par un utilisateur valide dans le système).

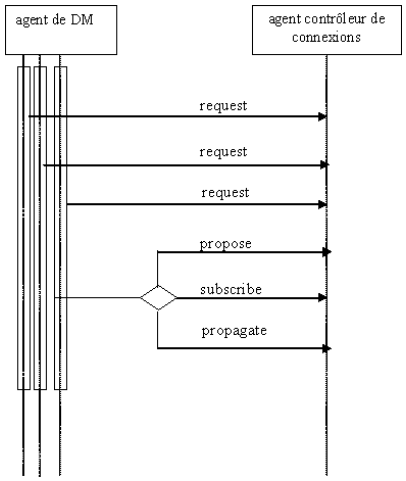


Figure 9.6. Exemple d'envoi concurrent de messages entre agents.

La Figure 9.7 présente un autre diagramme de séquence montrant la manière dont un *agent de SI* répond à l'envoi d'une requête par l'*agent de routage*. L'*agent de SI* envoie un message « *refuse* » s'il ne gère pas l'information répondant à la requête ; il envoie un message « *not understand* », s'il ne comprend pas la requête ; il envoie un message « *propose* » avec les résultats de cette requête.

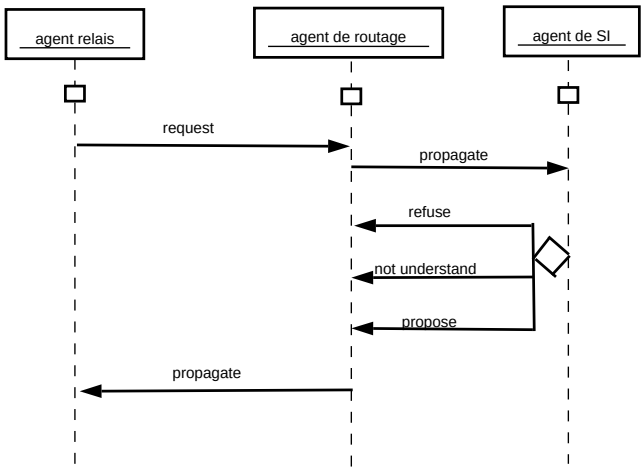


Figure 9.7. Exemple d'exécution conditionnée d'envoie de messages entre des agents.

La Figure 9.8 montre un autre *diagramme de séquence AUML* plus complexe qui représente les interactions entre les agents du *SMA de connexion* et du *SMA de communication* lorsqu'un *agent de DM* demande une information. Ce diagramme montre les messages échangés par les agents de *PUMAS* (spécifiquement, le chemin entre l'*agent de DM* et l'*agent relais*).

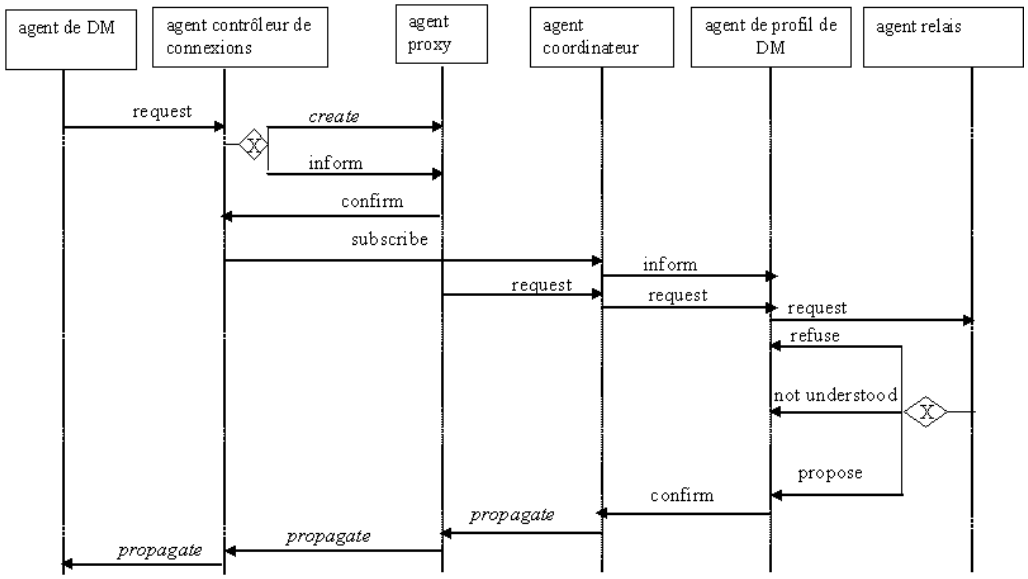


Figure 9.8. Diagramme de séquence AUML pour une requête d'information.

Le diagramme d'états groupant les états valides du traitement d'une requête est montré par la Figure 9.9.

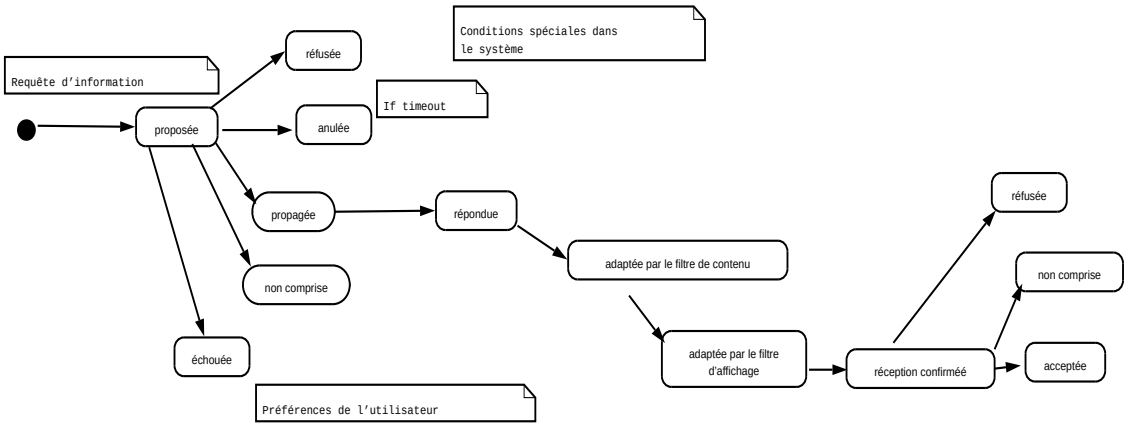


Figure 9.9. Diagramme d'état du traitement d'une requête.

9.1.2 Modélisation du Système de Gestion de Profil Contextuel

Dans cette section, nous montrons les diagrammes de classes qui représentent les trois types de *préférences* et le *profil contextuel de l'utilisateur*. Le diagramme de classes *UML* qui représente les *préférences d'activité, de résultat* et *d'affichage* est illustré par la Figure 9.10.

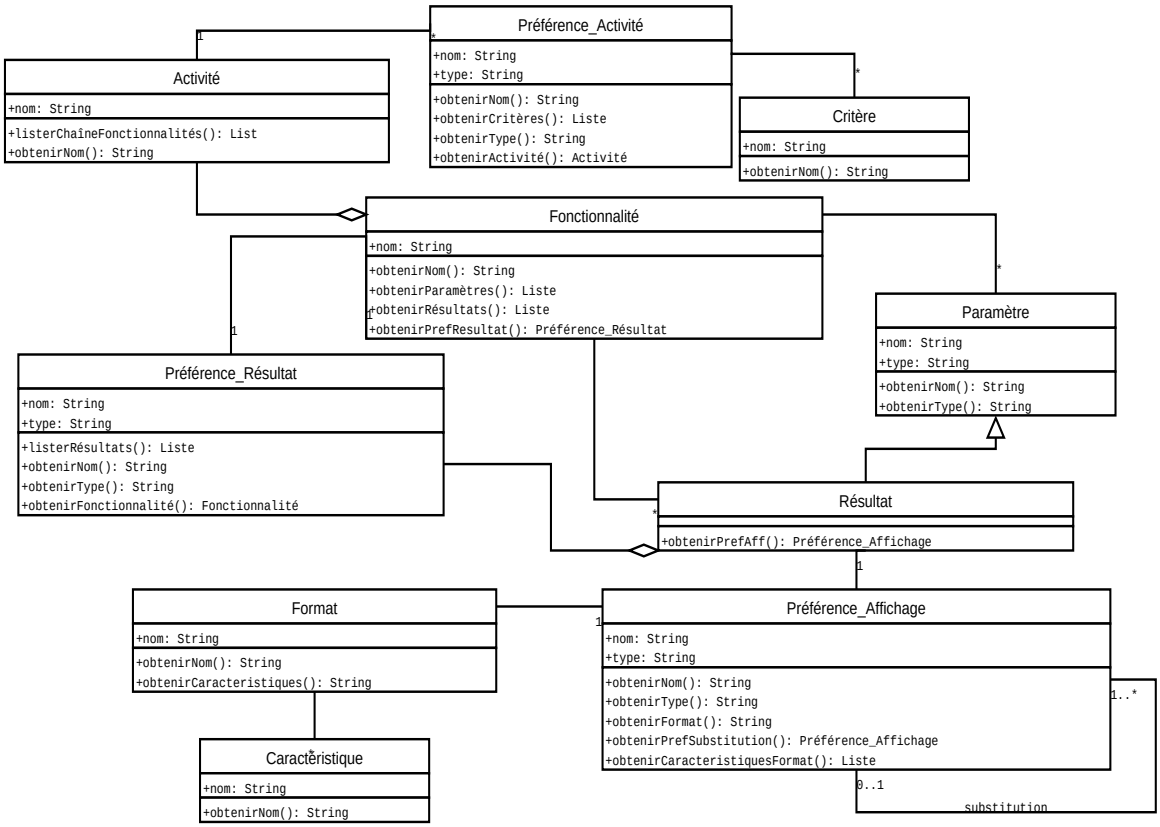


Figure 9.10. Diagramme de classes de préférences d'activité, de résultat et d'affichage.

Le diagramme de classes *UML* qui représente le *profil contextuel de l'utilisateur* et ses relations avec les *préférences de l'utilisateur* est illustré par la Figure 9.11 :

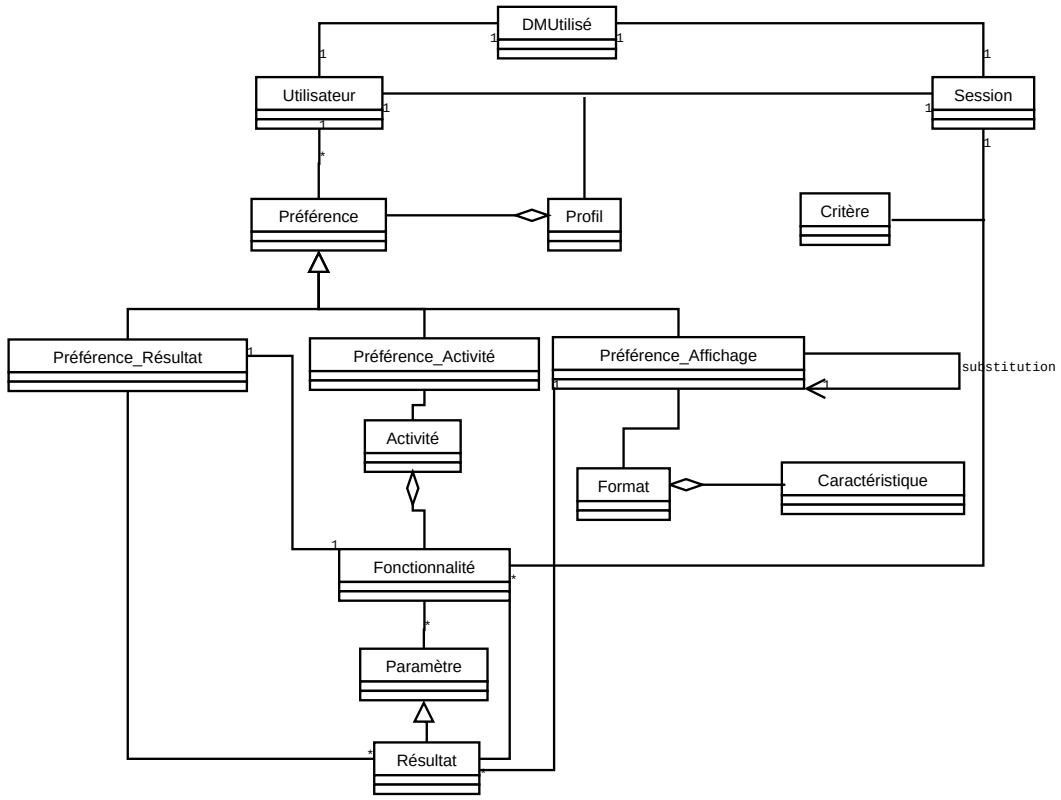


Figure 9.11. Diagramme de classes du profil contextuel d'un utilisateur.

Etant donné qu'un utilisateur peut accéder à un ou plusieurs *SIW* à travers son *DM* et qu'il peut accomplir des activités différentes lors de ses connexions au *SIW*, le *Système de Gestion de Profils Contextuels* utilise un ensemble de quatre fichiers écrits en *OWL*, stockés dans le *DM*, afin de définir le *profil contextuel de l'utilisateur* et d'adapter l'information. Ces fichiers contiennent les caractéristiques du *contexte d'utilisation* de la session en cours telles que les caractéristiques du *DM*, de la session, les *préférences de l'utilisateur*, et sa localisation. Nous expliquons chacun de ces fichiers ci-dessous :

- Le fichier *de session* contient les activités (et leurs fonctionnalités) d'une session, enregistrées au fur et à mesure qu'elles sont exécutées par le *SI*. Dans ce fichier, chaque fonctionnalité est représentée au moyen d'une liste (*fonctionnalité, paramètres, résultats* et *état*). L'état d'une *fonctionnalité* peut prendre pour valeur « *commencée* », « *finie* », « *suspendue* », *etc*. Ce fichier contient aussi l'information sur les caractéristiques de la session : l'utilisateur connecté, le temps de début de la session et le type de *DM* connecté.
- Le fichier *de localisation* contient la localisation *physique* et *logique* de l'utilisateur : ville, pays, rue, adresse *IP*, orientation, vitesse de déplacement.
- Le fichier *d'utilisateur* contient les *préférences de l'utilisateur*, enregistrées comme présentées dans la section 8.1.

– Le fichier de *dispositif* contient les caractéristiques du *DM* (telles que la taille de l'écran, la mémoire, la batterie) utilisé pour la connexion au système.

Les quatre fichiers sont définis en utilisant les extensions de *CC/PP* proposées par Indulska *et al*. [Indu03a].

9.1.3 Protégé comme outil pour définir les ontologies

Dans cette section nous présentons *Protégé*[88], un outil supportant la création, la visualisation et la manipulation d'ontologies, en les exportant sous plusieurs formats de représentations tels que *RDF*(S), *OWL*, et *XML* Schéma (cf. Figure 9.12). *Protégé* est un éditeur d'ontologies fournissant un environnement de « *plug-ins* » [89] (tels que ceux permettant l'utilisation de *RACER*[90], *JessTab*[91], *ezOWLTAb*, cf. Figure 9.13) qui facilitent le développement d'applications à bases de connaissances et la construction de modèles de domaine avec des ontologies.

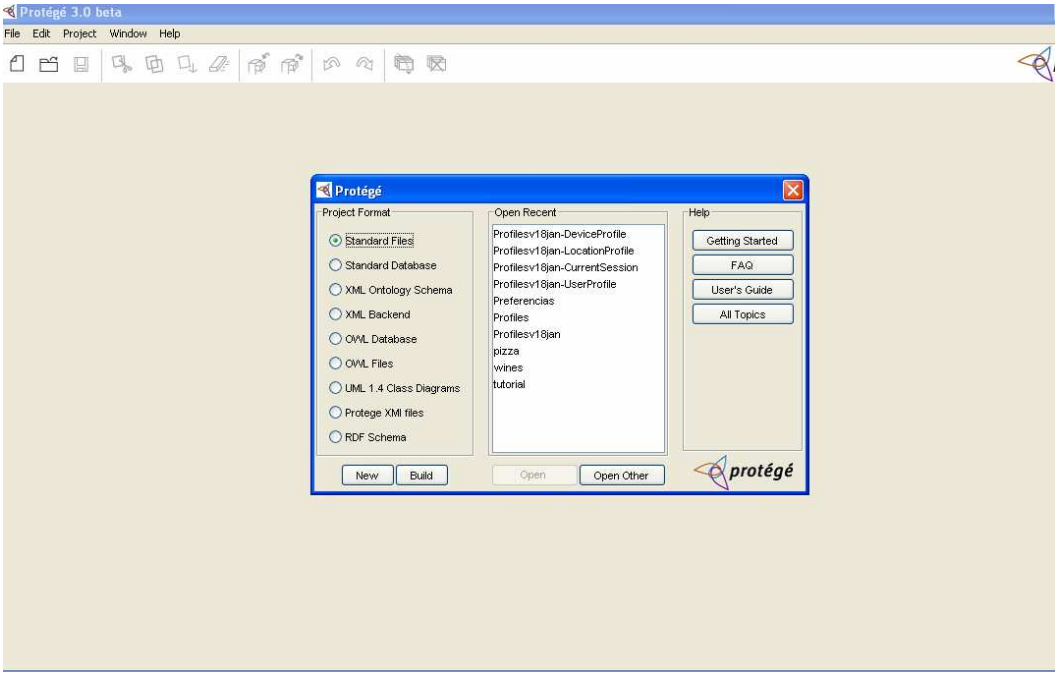


Figure 9.12. Interface de Protégé. Il est possible d'ouvrir un projet sous plusieurs formats (*cf. la liste des boutons radio*).

[88] http://protege.stanford.edu

[89] La liste complète des « *plug-ins* » de *Protégé* est disponible sur : *http://protege.cim3.net/cgi-bin/wiki.pl?ProtegePluginsLibraryByType*

[90] *Protégé* possède une interface pour communiquer avec Racer (http://www.racer-systems.com/), et un raisonneur pour classer des ontologies. Le raisonneur peut seulement classer des classes définies, c'est-à-dire, des classes avec au moins un ensemble de conditions nécessaires et suffisantes.

[91] http://www.ida.liu.se/~her/JessTab/

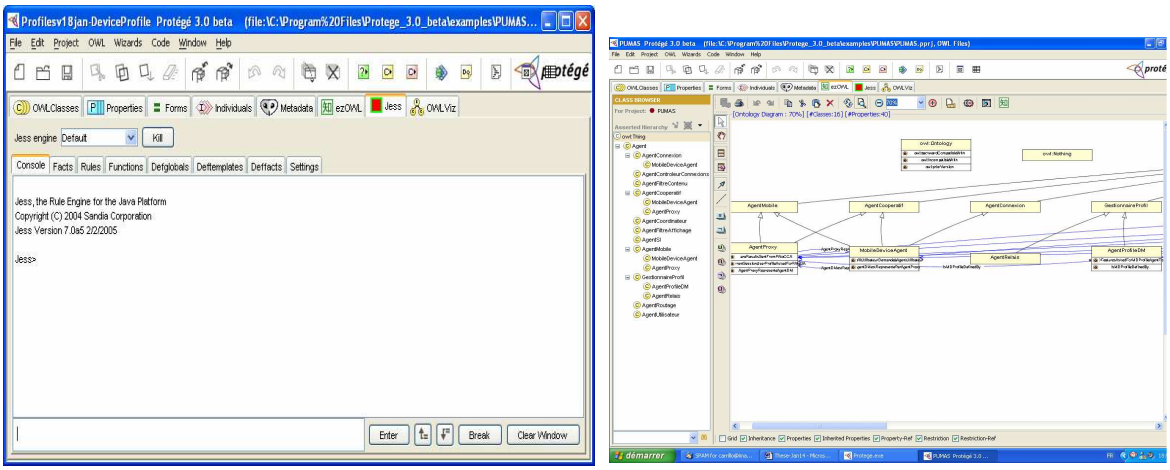


Figure 9.13. L'nterface de Protégé fournit différents plug-ins, par exemple, pour la gestion de connaissances (*JessTab à gauche*) *et pour la visualisation d'ontologies* (*ezOWL à droite*).

Pour *Protégé*, une *ontologie* décrit les concepts et relations pertinentes dans un domaine particulier, en fournissant un vocabulaire pour ce domaine et une spécification de la signification des termes utilisés dans ce vocabulaire. Protégé supporte deux manières de modéliser des ontologies en utilisant les éditeurs : « *Protégé-Frames* » et « *Protégé-OWL* ».

L'éditeur « *Protégé-Frames* » permet aux utilisateurs de construire des ontologies en utilisant le protocole « *Open Knowledge Base Connectivity* (*OKBC*)[92] ». Dans ce modèle, une *ontologie* est définie comme :

« *un ensemble de classes organisées dans une hiérarchie et représentant les concepts d'un domaine, un ensemble de « slots » associés aux classes afin de décrire leurs propriétés et relations, et un ensemble d'exemples de ces classes – exemples individuels des concepts avec des valeurs spécifiques pour leurs propriétés* ».

L'éditeur « *Protégé-OWL* » permet aux utilisateurs de créer des ontologies en utilisant le Web *Sémantique* en particulier *OWL*[93]. Pour *OWL*, une ontologie « *peut inclure des descriptions de classes, propriétés, leurs exemples* (« *instances* ») *et la sémantique associée entre concepts* ».

Protégé produit des fichiers *OWL* (en utilisant la version *OWL-Full*[94], cf. Figure 9.14) décrivant des concepts. Un concept peut être défini en fonction de concepts plus simples et il peut utiliser un ensemble plus riche d'opérateurs (« *and* », « *or* », « *negation* »). De plus, *Protégé* fournit un raisonneur pouvant vérifier si toutes les instructions et les définitions de l'ontologie sont mutuellement consistantes, et pouvant aussi reconnaître la définition qui s'adapte mieux à un concept d'après son usage.

---

[92] *OKBC* : Open Knowledge Base Connectivity. http://www.ai.sri.com/~okbc/
[93] http://www.w3.org/TR/owl-guide/
[94] http://www.w3.org/TR/owl-features/.

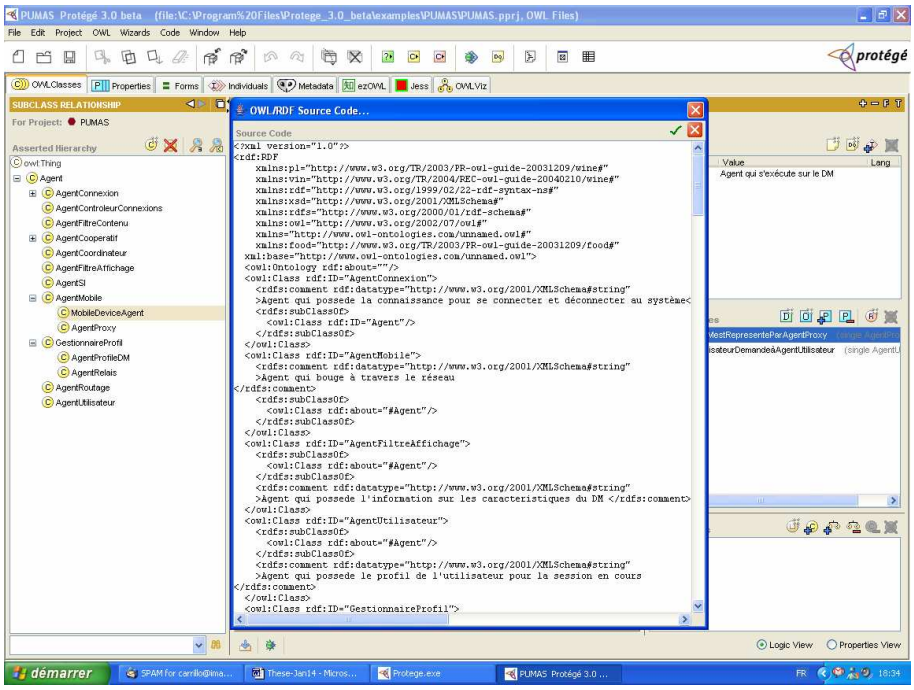


Figure 9.14. Interface de Protégé. Génération du code d'une ontologie en OWL.

Les composants des ontologies *OWL* utilisés en Protégé sont les « *Individuals* » et les « *Properties* ». Les « *Individuals* » (*Protégé Instances*) représentent des objets dans un domaine spécifique. Cependant, *OWL* n'utilise pas le « *Unique Name Assumption* (*UNA*) », donc, deux noms différents peuvent faire référence au même « *individu* ». Les « *Properties* » (*Protégé Slots*) sont des relations binaires ou individuelles. Elles correspondent aux rôles dans la logique de description et aux attributs et liens en représentations de connaissances par objets. Une propriété possède un domaine et une image (cf. Figure 9.15). Une propriété peut se dire :

- *Inverse :* si elle représente un changement du sens d'une propriété existante. Par exemple, pour la propriété « *AgentDMestRepresenteParAgentProxy* », sa propriété inverse est « *AgentProxyRepresenteAgentDM* » (cf. Figure 9.15) ;
- *Transitive :* une propriété *P* est transitive, si dans le cas où *P* lie l'individu « *a* » à l'individu « *b* », et l'individu « *b* » à l'individu « *c* », alors nous pouvons inférer que l'individu « *a* » est lié à l'individu « *c* » via la propriété *P*. Par exemple, si Angela *aCommeAncêtre* LuzMila et LuzMila *aCommeAncêtre* Inés, alors Angela *aCommeAncêtre* Inés. Si une propriété transitive définit une propriété inverse, cette propriété inverse est aussi transitive ;
- *Symétrique :* une propriété *P* est symétrique, si la propriété qui lie l'individu « *a* » à l'individu « *b* », lie aussi l'individu « *b* » à l'individu « *a* ». Par exemple si Angela *estMariéAvec* Fernando, alors Fernando *estMariéAvec* Angela ;

– *Fonctionnelle* : une propriété *P* est fonctionnelle s'il existe au moins un individu qui est lié à un autre individu à travers *P*. Par exemple, l'*agent de DM* est lié à l'*agent proxy* par la propriété « *connexion* ».

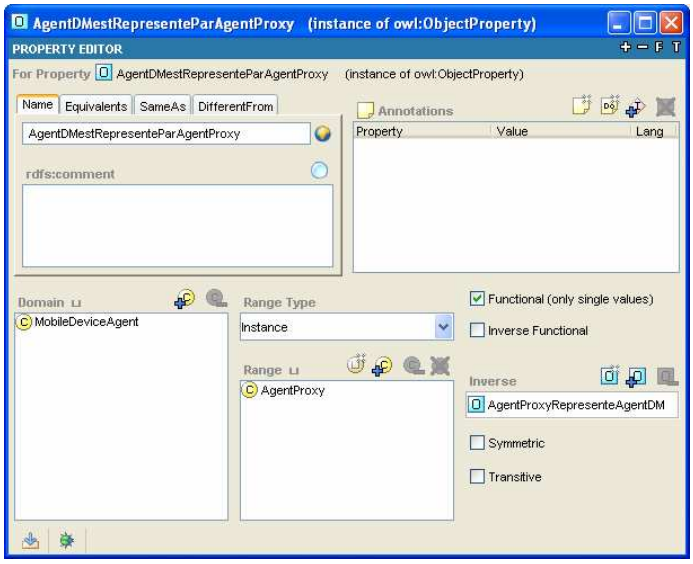


Figure 9.15. Interface de Protégé. Création d'une propriété.

OWL a trois types de propriétés :

– *Object properties :* une telle propriété associe un individu à un autre individu. Par exemple, un *agent proxy* représente un *agent de DM* (cf. Figure 9.15) ;

– *Datatype properties :* ces propriétés associent un individu à un *XML Schéma Datatype* value ou à un *rdf literal*. Par exemple, l'agent s'exécutant sur un *DM* situé à moins de "2^^xsd:integer" Kms ;

– *Annotation properties* : Ces propriétés sont utilisées pour ajouter de l'information additionnelle (métadonnées) à des classes, individus et propriétés object/datatype.

Les propriétés possèdent des restrictions qui peuvent être classifiées en :

– *Conditions Nécessaires* : si toutes les conditions de la classe sont « *nécessaires* », cette classe est nommée *Primitive.* « *Si un individu est membre d'une classe primitive alors, il est nécessaire qu'il remplisse ses conditions* ». C'est-à-dire, « *si un individu remplit les conditions d'une classe primitive, alors il peut être membre de cette classe* ».

– *Conditions nécessaires et suffisantes* : une classe ayant un ensemble de conditions nécessaires et suffisantes, est connue comme une classe *Définie*. « *Si un individu est un membre d'une classe définie alors il doit satisfaire ses conditions* ».

Les classes n'ayant que des conditions nécessaires sont connues comme classes *Partielles* (par exemple, la classe « *AgentDM* », cf. Figure 9.16). Les classes ayant au moins un ensemble de conditions nécessaires et suffisantes sont des classes *Complètes*.

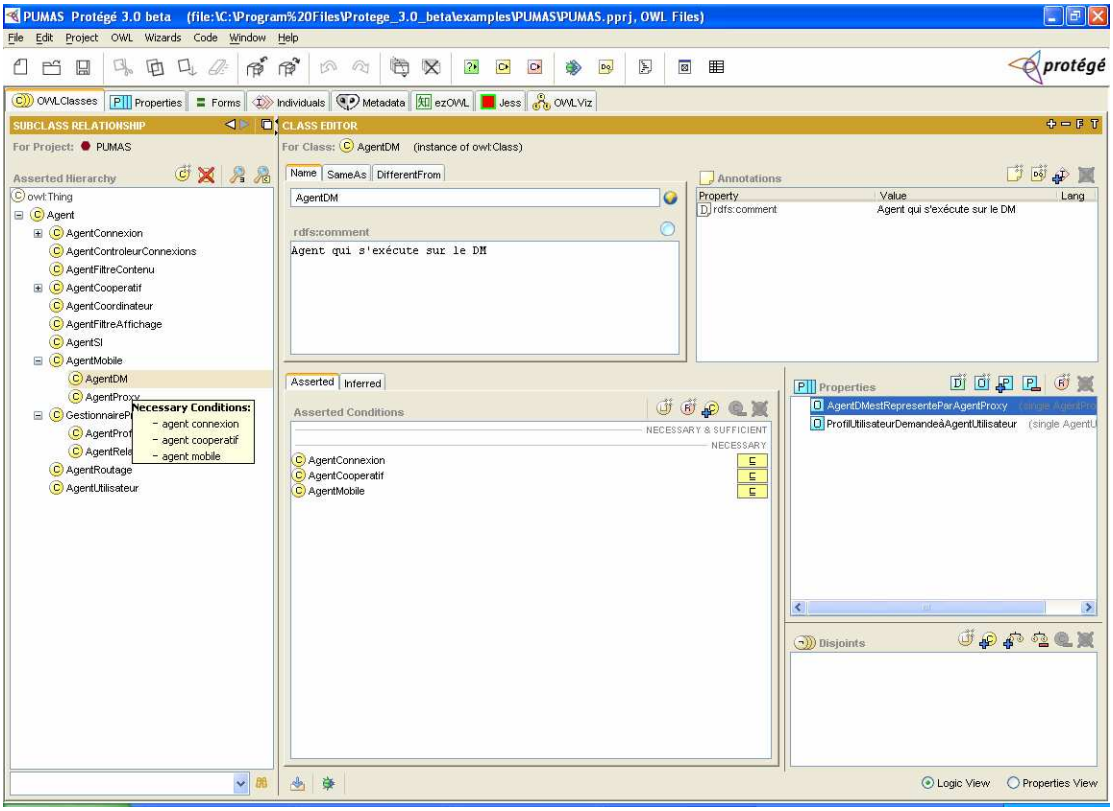


Figure 9.16. Interface de Protégé. Définition de la hiérarchie de classes et leurs propriétés.

9.1.4 Plates-formes de développement

Dans cette section, nous présentons tout d'abord les spécifications *FIPA* pour des plates-formes de développement et ensuite quelques exemples de ces plates-formes.

9.1.4.1 Spécifications *FIPA* pour les applications nomades et les plates-formes de développement

La *FIPA* est une organisation chargée de la production et de l'évaluation de spécifications pour les *SMA*. Son but est de garantir l'interopérabilité entre *SMA* développés par différentes compagnies et organisations. Une spécification ayant un grand nombre de mises en œuvre et présentant une contribution importante, peut devenir un standard *FIPA*. En particulier pour les applications nomades basées sur des agents, la *FIPA* a proposé différentes spécifications[95] et un modèle de référence possédant trois composants : *i)* le « *Monitor Agent* » qui est le responsable du contrôle de la qualité de la transmission de messages ; *ii)* le « *Control Agent* » qui est chargé de sélectionner le *Protocole de Transport de Message* (*MTP*) et de contrôler les connexions au

[95] Les spécifications pour les applications nomades sont : [FIPA00014], [FIPA00069], [FIPA00088] et [FIPA00094].

système ; *iii)* l'agent « *middleware* » qui implémente un *Protocole de Transport de Message* afin d'assurer la *connexion et le transport des messages*.

Pour les plates-formes, la *FIPA* a défini les spécifications suivantes[96] décrites par Dale [Dale00] (cf. Figure 9.17) :

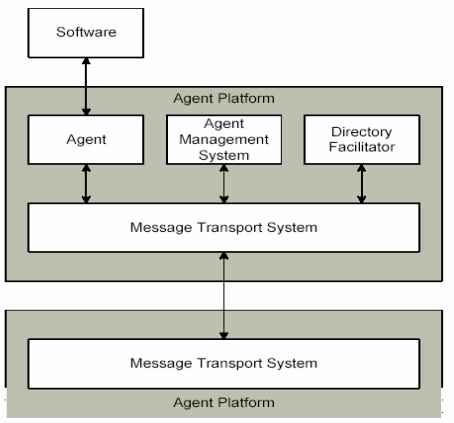


Figure 9.17. Modèle de Référence pour la gestion des agents (*d'après* [*fipa00023*]).

- *Agent :* il s'agit d'un logiciel qui est le noyau d'une application. Il communique avec d'autres agents en utilisant un *Agent Communication Langage* (*ACL*) ;
- *Directory Facilitator* (*DF*) : il s'agit d'un composant optionnel pour une plate-forme d'agents. Il est chargé d'offrir le service de pages jaunes aux agents. Ces derniers inscrivent leurs services dans ces pages. Le *Directory Facilitator* accomplit le rôle d'*Agent Directory Service* (*ADS*). Selon la spécification [FIPA00001], l'*Agent Directory Service* fournit un espace pour que les agents enregistrent leurs descriptions comme « *agent-directory-entries* » et qu'ils puissent rechercher d'autres agents avec lesquels ils voudraient interagir ;
- *Agent Management System* (*AMS*) : il s'agit d'un composant obligatoire de la plate-forme qui gère son utilisation. Il offre aux agents le service de pages blanches où ils enregistrent leurs adresses et identificateurs dans le système ;
- *Message Transport Service* (*MTS*) : il s'agit de la méthode de communication entre les agents ;
- *Software* : il fait référence aux logiciels, en dehors de la plate-forme, utiles pour que les agents puissent accomplir une tâche ;
- Une *plate-forme d'agents* (*PA*) : elle fournit l'infrastructure physique pour l'exécution des agents. Elle est composée de la machine, du système d'exploitation, des logiciels pour supporter les agents, des composants *FIPA* pour la gestion des agents (*Directory*

[96] http://www.fipa.org/specs/fipa00023/SC00023J.pdf

Facilitator, *Agent Management System* et *Message Transport Service*) et les agents mêmes.

9.1.4.2 Plates-formes conformes aux spécifications de la FIPA

Parmi les implémentations de plates-formes conformes aux spécifications de la *FIPA*, on peut citer[97] : *Agent Development Kit*, *April Agent Platform*, *Comtec Agent Platform*, *FIPA-OS*, *Grasshopper*, *JACK Intelligent Agents*, *JADE*, *JAS* (*Java Agent Services API*), *LEAP*, *ZEUS*. Nous décrivons brièvement chacune de ces plates-formes avec ses caractéristiques et d'une manière plus détaillée la plate-forme *JADE*. Un tableau comparatif des plates-formes est donné dans l'annexe 3[98].

a – Agent Development Kit

Il est développé par Tryllian BV[99]. Le Tryllian Agent Development Kit (ADK) permet de concevoir et de mettre en œuvre des applications réparties, basées sur la technologie des agents mobiles. Il fournit un grand nombre de fonctions qui permettent aux développeurs de créer et de définir des agents et leur comportement. Il offre des possibilités de construction de logiciels d'une manière rapide, grâce aux bibliothèques standard de comportements d'agents. A présent, Tryllian propose la version 3.2.0 sous licence *LGPL*. Les composants de cette plate-forme sont d'une part, l'*Agent Foundation Classes* (*AFC*) et d'autre part, l'*Agent Runtime Environment* (*ARE*). L'*Agent Foundation Classes* permet aux développeurs de définir les composants d'une application basée sur des agents, leurs interrelations et les tâches de chacun constituant leur comportement. L'*Agent Runtime Environment* met en œuvre toutes les communications entre les agents et le système. Il est composé d'un « *habitat* » contenant des places. Une place est un ensemble d'agents.

b – April Agent Platform (AAP)

AAP est un projet dirigé par Jonathan Dale et Francis G. McCabe[100]. Actuellement, les *Fujitsu Laboratories of America* ont les droits sur *AAP*, écrit en *APRIL*[101] et conforme au modèle *ICM*[102]. *AAP* constitue un environnement d'exécution pour des agents. Ses composants sont des agents, un mécanisme de pages jaunes (*Directory Facilitator*) et de pages blanches pour les services de résolution de noms (*Agent Management System*) et un *MTS* (*Message Transport System*).

Les versions d'April 4.4.3 et AAP 0.9.0 sont disponibles sous Linux, Unix, Solaris et Windows sous licence GNU General Public License (GPL), GNU Library ou Lesser General Public License (LGPL).

---

[97] http://www.fipa.org/specs/fipa00014/SI00014H.html
[98] http://www-lsr.imag.fr/users/Angela.Carrillo/
[99] http://www.tryllian.com/technology/product1.html
[100] http://sourceforge.net/projects/networkagent/
[101] *APRIL* : Agent Process Interaction Language
[102] *ICM* : InterAgent Communication Model

c – Comtec Agent Platform

Comtec Agent Platform [Sugu99] a implémenté les spécifications FIPA 97 pour la gestion d'agents, leur langage de communication et le service d'ontologie. La dernière version date de 1999, après le transfert de Comtec Corporation vers Communication Technologies. Il existe des versions gratuites sous Windows et Unix.

d – FIPA-OS

FIPA-OS a été la première plate-forme de libre utilisation de la technologie *FIPA*, aussi bien dans le milieu académique que dans l'industrie. *FIPA-OS* signifie « *Foundation for Intelligence Physical Agents – Open Source* » et les droits réservés appartiennent à *NORTEL NETWORKS Corporation, Emorphia* et aux « *contributors* ». Cette plate-forme est développée sous *JAVA* et il existe des versions pour Windows, Unix et Solaris. Une version FIPA-OS pour PDA, appelée microFIPA-OS, est disponible sous les systèmes d'exploitation UNIX et Pocket PC.

e – Grasshopper

Grasshopper [Mage99] est une plate-forme d'exécution et de développement d'agents mobiles, qui représente un environnement de traitement réparti. Elle a été créée par GMD FOKUS (German National Research Center for Information Technology et Research Institute for Open Communication Systems). Elle est conforme aux standards de *FIPA* et *MASIF*[103].

Grasshopper est composée de régions, de places, d'agences et de différents types d'agents (agents mobiles et stationnaires). La région est responsable de la gestion de tous les composants (les agences, les agents, les autres places). Une place est un ensemble de fonctionnalités d'une agence. L'agence est l'environnement d'exécution des agents et elle les groupe selon un critère défini. Elle est aussi responsable de l'enregistrement de tous ses agents et de leurs mouvements dans le Region Registry. L'agence est composée de deux parties : d'une part, le « *Core agency* » qui représente la fonctionnalité minimale des agences, et d'autre part, une ou plusieurs places [Mage99].

Il existe des versions sous Windows et Unix. Tout logiciel développé en utilisant Grasshopper doit être déclaré comme « *freeware* ». La première version a été développée avec les spécifications d'OMG MASIF, à laquelle IKV++ Technologies AG a apporté des améliorations. Une version nommée Grasshopper Micro Edition a été créée pour les dispositifs mobiles. Grasshopper supporte deux mécanismes de sécurité : d'une part, la sécurité externe utilisant le certificat X.509 et le protocole *SSL*, et d'autre part, la sécurité interne au niveau des agences, basée sur les mécanismes de sécurité de Java.

---

[103] *MASIF* : *Mobile Agent System Interoperability Facility*. C'est le premier standard du OMG (*Object Management Group*) pour la définition des relations et la coordination des systèmes multi-agents. http://www.omg.org/docs/orbos/98-04-05.pdf

f – JACK Intelligent Agents

JACK est un environnement basé sur des composants écrits en *JAVA* dont le but est la construction, l'exécution et l'intégration de systèmes à agents[104]. Il est développé par l'*Agent Oriented Software Group* (*AOS*). Ce groupe offre une version commerciale et une version d'évaluation (la version 5.2 est disponible pour 60 jours[105]). Il existe des versions pour Windows, Solaris, Macintosh et Unix. Son langage de programmation est une extension de *JAVA* réunissant les concepts d'agents, bibliothèques fonctionnelles d'agents, événements, plans, bases de connaissances, gestion de ressources et de concurrence, entre autres. Il utilise le modèle de sécurité de *JAVA* et les agents peuvent conserver leur information privée (qui ne peut pas être accédée par d'autres).

JACK fournit des outils graphiques pour la conception et la construction d'applications : « *design tool* » (conception de l'application qui, à partir de diagrammes, peut générer un code source) et « *plan editing tool* » (définition de plans pour les tâches et les activités d'agents, possibilité de suivre graphiquement l'exécution du plan). Ses composants sont un environnement de développement et d'exécution, un compilateur, les modèles d'agents *BDI*[106] et les équipes d'agents.

g – JAS (*Java Agent Service API*)

C'est la *JAVA Specification Request* (*JSR* 87[107]) qui définit un ensemble d'objets et d'interfaces pour supporter l'exécution d'agents communicatifs. « *Java Agent Service* » est un travail collectif d'entreprises comme Fujitsu Limited, HP, IBM, SUN MicroSystems, University of West Florida Institute of Human-Machine Cognition, entre autres.

JAS est composée, d'une part, de classes *JAVA* correspondant à des éléments comme *ACL*, à des agents *FIPA* et leurs descriptions et, d'autre part, d'interfaces *JAVA* offrant des services comme l'enregistrement d'agents et les communications (similaire au *Java Message Service – JMS* - et utilise son modèle de sécurité pour l'envoi et la réception de messages). Elle est considérée comme une interface de spécification.

h – ZEUS

Selon la *FIPA*, *ZEUS* est un « *Open Source toolkit* » développé par BT LABS[108], pour la construction d'applications collaboratives basées sur des agents. Il offre des bibliothèques pour des agents (composants logiciels qui ont des fonctionnalités des systèmes multi-agents) et des outils aussi bien pour le développement (descriptions des agents, leur communication, leurs

---

[104] http://www.agent-software.com/shared/products/index.html

[105] http://www.agent-software.com/shared/extrafiles/regform.html

[106] *BDI* : *Belief, Desire, Intention.* C'est une modélisation de la capacité de raisonnement des agents et leur comportement aux changements du contexte où ils agissent. Dennet, D.C., The Intentional Stance, The MIT Press, MA. [Coll03]

[107] http://www.jcp.org/en/jsr/detail?id=087

[108] http://more.btexact.com/projects/agents.htm

relations, leurs tâches, la définition des ontologies à utiliser, *etc*.) que pour la visualisation d'applications (pour l'affichage d'agents, des statistiques, l'exécution de tâches).

La version 2.0 de *ZEUS* est un Open Source Software[109], gratuit pour la recherche et l'industrie sous la licence Mozilla Public Licence. Il existe deux versions, une pour Windows et l'autre pour Solaris.

i – JADE

JADE (***J****ava* ***A****gent* ***DE****velopment Framework*) est un « *framework* » développé en *JAVA*. Il offre des outils graphiques pour le développement et la mise en œuvre d'applications basées sur des agents. La configuration logicielle minimale requise est la version 1.4 de JAVA (JRE ou JDK).

Une association entre la plate-forme *JADE* et les bibliothèques *LEAP* existe. Cette association permet l'utilisation de JADE-LEAP et la compatibilité avec les environnements *JAVA* pour des *DM* (J2ME-CLDC MIDP).

JADE est un logiciel libre et il est distribué par *TILAB*[110] sous licence *LGPL* (*Lesser General Public License Version 2*). A présent, la dernière version est la 3.4.1 sortie en novembre 2006.

Ses composants sont un environnement d'exécution des agents, une bibliothèque de classes pour développer les agents, et des outils graphiques pour contrôler et gérer l'exécution des agents.

Pour suivre les spécifications FIPA, JADE offre : un AMS (*Agent Management System*) avec un registre automatique des agents, un DF (*Directory Facilitator*) et un ACC (*Agent Communication Channel*).

JADE est une plate-forme répartie d'agents. Elle peut diviser l'agent en plusieurs serveurs (sans « firewalls » entre eux). Chaque serveur exécute la JVM et l'application basée sur des agents (implémentés comme « threads » et avec un GUID – *Globally Unique Identifier*).

JADE offre des outils graphiques pour la gestion des agents : *Dummy Agent* pour envoyer et recevoir des messages *ACL* (« *Agent Communication Language* ») et le *Sniffer Agent* pour contrôler les communications entre plusieurs agents.

Ses paquets sont [Bell06] :

- *Jade.core* : c'est le noyau du système. Il possède la classe « *Agent* » permettant de définir les tâches et le comportement des agents ;
- *Jade.lang.acl* : cet ensemble de classes permet d'utiliser *ACL* comme mécanisme de communication d'agents. Chaque message *ACL* est encapsulé comme un objet *JAVA* (String) ;

[109] http://sourceforge.net/projects/zeusagent/
[110] http://jade.tilab.com/

- *Jade Content* : c'est un ensemble de classes utiles à la définition d'ontologies par l'utilisateur. Il utilise *JESS* pour définir la connaissance de l'agent ;
- *Jade.domain :* cet ensemble de classes supporte les mécanismes de « l'*Agent Management System* » et du « *Directory Facilitator* » pour la gestion du cycle de vie des agents, des services de pages jaunes et blanches, et la définition des concepts liés aux ontologies ;
- *Jade.gui :* des classes liées à la conception et la gestion d'interfaces graphiques ;
- *Jade.mtp :* il fournit une interface qui doit être implémentée par l'application selon le *Message Transport Protocol* choisi. Cette interface est utilisée pour l'envoi et la réception de messages, et pour la connexion avec différentes plates-formes ;
- *Jade.proto* : des classes permettant la définition des protocoles d'interaction standard de la spécification *FIPA* comme fipa-request, fipa-query, fipa-contract-net, *etc.* ;
- *Jade.wrapper : JADE* s'utilise comme une bibliothèque qui permet aux applications de définir leurs agents et les conteneurs de ces agents.

j – LEAP

LEAP[111] est l'acronyme de « *Lightweight Extensible Agent Platform* », plate-forme pour des applications basées sur des agents dont l'accès est fait à travers des dispositifs mobiles tels que *PDA* et téléphones portables. Des entreprises comme *Motorola*, *ADAC*, *BT*, *BROADCOM*, *Telecom LAB* (Italie), *SIEMENS*, entre autres, participent à son développement.

Elle a été développée en *JAVA* comme une bibliothèque qui doit être utilisée avec *JADE* et peut s'exécuter en versions standard de *JAVA* comme J2SE, J2ME, pJAVA (JDK 1.1.x). Sa licence est *LGPL*.

9.1.5 Correspondance entre les composants de PUMAS et les composants JADE

Afin d'implémenter des applications au dessus de *PUMAS*, nous avons choisi *JADE-LEAP* [JADE06]. Nous l'avons testé en utilisant deux types différents de *DM* : des Pocket PC avec Windows CE, en utilisant Crème – kVM qui est compatible avec Personal Java, et des *PDA* avec *PalmOS* en utilisant une implémentation *MIDP* 1.0. Après ces tests, nous avons trouvé quelques correspondances entre les composants de *PUMAS* et ceux de *JADE-LEAP*.

Nous pouvons associer :

- l'*AMS* (*Agent Management System*) de *JADE-LEAP* à notre *agent coordinateur ;*
- le *DF* (*Directory Facilitator*) à notre *agent de profil de DM*, l'*agent relais* et l'*agent de routage* ;
- l'*ACC* (*Agent Communication Channel*) à notre *agent contrôleur de connexions*.

[111] http://www.fipa.org/resources/livesystems.html#LEAP

JADE-LEAP propose aussi un *Agent Dummy* pour envoyer et recevoir les messages *ACL* (*Agent Communication Languages*), et un *Agent Sniffer* pour contrôler les communications entre les agents. Pour la représentation de connaissances, *JADE-LEAP* utilise *JESS* et un ensemble de classes (*JADE Content package*) définissant des ontologies. *JADE-LEAP* permet aussi à l'utilisateur de définir ses propres ontologies. Lors de nos tests des exemples fournis avec *JADE-LEAP*, nous avons trouvé quelques problèmes tels que le manque de correspondance entre les agents actifs listés dans la plate-forme et les agents connectés (quelques agents déconnectés apparaissent comme connectés pour la plate-forme). Pour cette raison, nous proposons l'*agent contrôleur de connexions* (qui contrôle tous les agents connectés) et l'*agent coordinateur* (ayant un registre des agents du système et qui pourrait jouer le rôle d'*agent contrôleur de connexions* si le *contrôleur* échoue). Nous n'avons pas pu exécuter *JADE-LEAP* d'une manière « *stand-alone* » sur notre *Pocket PC*. Nous avons utilisé une exécution « *split execution* » qui simule une *architecture P2P hybride* avec les avantages suivants : connaissance des agents actifs, de leurs services et de leurs capacités; moins de trafic sur le réseau, un mécanisme de registre (souscription) et d'authentification permettant à un agent de reconnaître tous ses pairs.

9.2. Exemples d'interfaces fournies à l'utilisateur/administrateur

Cette section présente la manière dont *PUMAS* fournit à l'utilisateur différentes interfaces pour :

1. accéder à la plate-forme ;
2. connaître les agents connectés ;
3. connaître les agents déconnectés ;
4. envoyer des messages vers des agents spécifiques et/ou en *broadcast*.

9.2.1 Connexion de l'utilisateur à PUMAS

Nous avons réutilisé la version pour personal Java de l'exemple de « chat » fourni par la plate-forme *JADE-LEAP*. Pour les tests, nous avons utilisé des *Pocket PC* (*iPAQ* 5550). Dans la gestion des agents « *connectés* » à la plate-forme, nous avons identifié les problèmes suivants dont la plupart ont été résolus à l'aide des agents de *PUMAS* :

- Un même utilisateur n'a pas le droit d'initialiser plus d'une session ;
- Un agent déconnecté ne peut pas se reconnecter ;
- Les utilisateurs apparaissaient encore connectés sur la plate-forme *JADE-LEAP* après s'être déconnectés. *PUMAS* résout ce problème en utilisant l'*agent contrôleur de connexion* (pour cette application, nous avons créé une instance de cet agent appelée « *preferencemanager* » sur la plate-forme). Cet agent gère une liste d'agents connectés et une liste d'agents déconnectés ;
- A défaut d'une séparation de fonctionnalités de l'interface (*Front-End*) et des fonctionnalités de traitement de l'information propres d'un agent (*Back-End*), l'utilisateur de l'application sur le *Pocket PC* ne pouvait pas la quitter. L'application

restait toujours en exécution, mais sans fournir ses services. La solution fournit par *PUMAS* à ce problème est d'attribuer les fonctionnalités de l'interface à un *agent de DM* et celles de l'application à un *agent proxy*. D'une part, nous avons re-développé l'interface liée à l'*agent de DM* en donnant la possibilité à l'utilisateur de quitter l'application. D'autre part, nous avons créé pour l'*agent contrôleur de connexions* la fonctionnalité d'envoyer à l'*agent proxy* (associé à cet *agent de DM*) la liste des agents déconnectés ;

– Un seul utilisateur pouvait se connecter sur le Pocket PC. Le fait de séparer les fonctionnalités du *Back-End* et du *Front-End* (cf. ci-dessus) a aidé aussi à la résolution de ce problème. Les nouvelles interfaces des *agents de DM* sont plus légères (possèdent moins de composants graphiques) ;

– Un utilisateur pouvait envoyer des messages aux utilisateurs déconnectés. Dans ce cas, les erreurs produites ne s'affichaient pas sur l'interface de l'utilisateur mais elles s'affichaient sur la plate-forme. L'application de l'utilisateur et la plate-forme arrêtaient leur exécution. Le fait de gérer sur l'*agent contrôleur de connexions* les listes d'agents connectés et déconnectés, permet à chaque *agent proxy* de connaître vraiment les autres *agents proxy* connectés. Ces agents connectés sont les seuls destinataires potentiels des messages. Chaque *agent proxy* envoie à son *agent de DM* associé, sa liste d'agents connectés pour l'afficher sur l'interface du *DM* ;

– Les messages ne pouvaient pas être dirigés vers un ensemble d'utilisateurs. Ils pouvaient être transmis soit à un utilisateur particulier, soit à tous les utilisateurs (en *broadcast*). Pour résoudre ce problème, du côté de l'*agent de DM*, nous avons modifié son interface en permettant de sélectionner un ensemble de destinataires dans sa liste d'agents connectés. L'*agent de DM* compose et envoie un nouveau message qui contient la liste des destinataires plus le message original. Du côté de l'*agent proxy*, nous avons étendu les fonctionnalités concernant le traitement des messages reçus de son *agent de DM* et l'envoi des messages à un ensemble des destinataires.

Après le développement de notre application pour les utilisateurs, les problèmes précédents ont été résolus sauf les deux premiers (qui concernent directement la plate-forme *JADE-LEAP*).

Afin d'exécuter notre application, il faut démarrer la plate-forme *JADE-LEAP* (s'exécutant sur le serveur, cf. Figure 9.18). Quatre agents apparaissent : le *rma*[112], l'*ams*, le *df*[113] (cf. section 9.1.4.2i –) et le *preferencemanager* (notre *agent contrôleur de connexions*). Ce dernier est le gestionnaire de préférences et de connexions de *PUMAS* s'exécutant sur la plate-forme.

[112] rma : acronyme de « *Remote Management Agent* », est un agent qui agit comme une console graphique pour contrôler et gérer la plate-forme.
[113] Les agents *rma*, *ams* et *df* appartiennent à la plate-forme *JADE-LEAP* (cf. section 9.1.4.2i –).

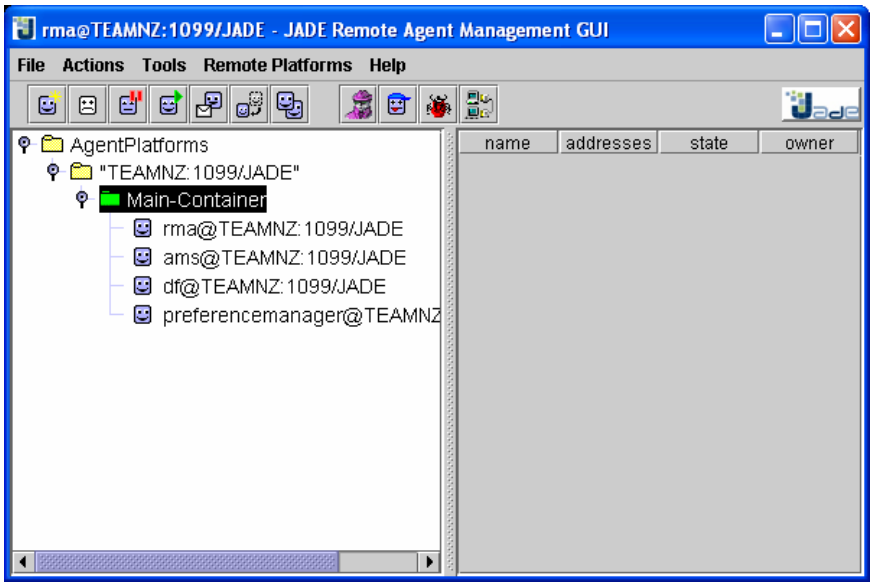


Figure 9.18. Plate-forme JADE-LEAP avec les agents de base de la plate-forme : rma, ams et df, et l'agent preferencemanager (*agent contrôleur de connexions*) *qui gère les connexions à PUMAS et les préférences de l'utilisateur.*

Pour les tests de connexion, nous créons trois *agents de DM* (en simulant trois utilisateurs qui se connectent à travers des *DM*) : Raphaël, Windson et Fernando (cf. Figure 9.19). Nous montrons ci-dessous les interfaces correspondant aux agents s'exécutant sur le serveur (*agents proxy*) et sur le *Pocket PC* (*agents de DM*). Une instance de cette interface permet à un utilisateur de se connecter à *PUMAS* et de connaître ses pairs (les utilisateurs connectés auxquels il peut envoyer des messages).

L'interface de connexion d'un utilisateur est composée d'un champ texte dans lequel, un utilisateur peut écrire le(s) nom(s) du (des) utilisateur (s) destinataire(s) d'un message. L'utilisateur peut écrire directement le(s) nom(s) des destinataires ou le(s) sélectionner à partir de la liste de participants (la fenêtre de « Pairs de »).

Un second champ texte permet à l'utilisateur de saisir son message.

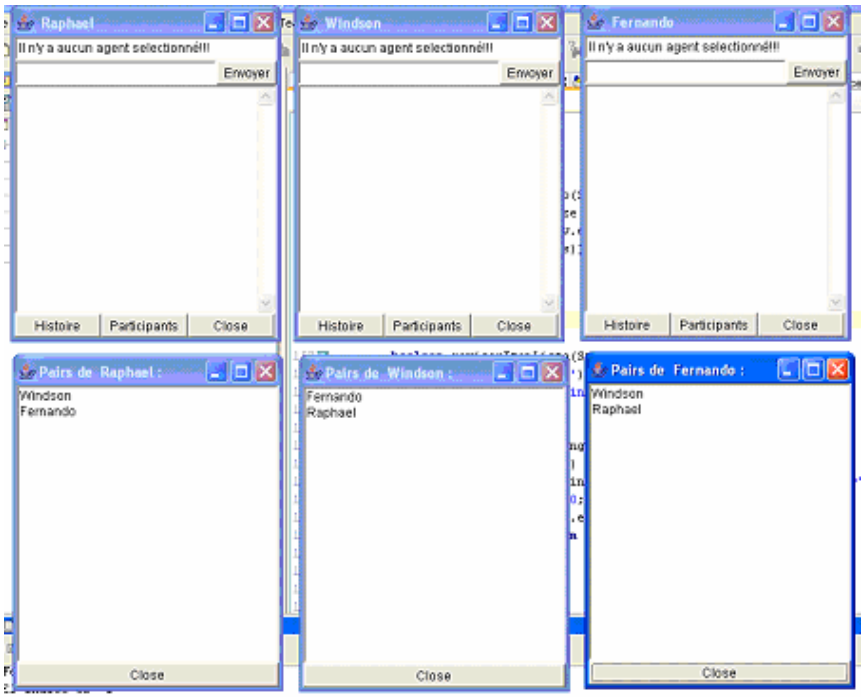


Figure 9.19. Interfaces de trois utilisateurs simulés, chacun avec un agent de DM associé. La simulation a été développée avec JBuilder. Les frames supérieurs montrent l'interface de chaque utilisateur, les inférieurs correspondent aux utilisateurs pairs de chacun.

Une fois que les utilisateurs se sont connectés, ils sont enregistrés dans la plate-forme comme des agents proxy et ils apparaissent à travers la création d'un Back-End[114] qui leur est dédié (cf. Figure 9.20).

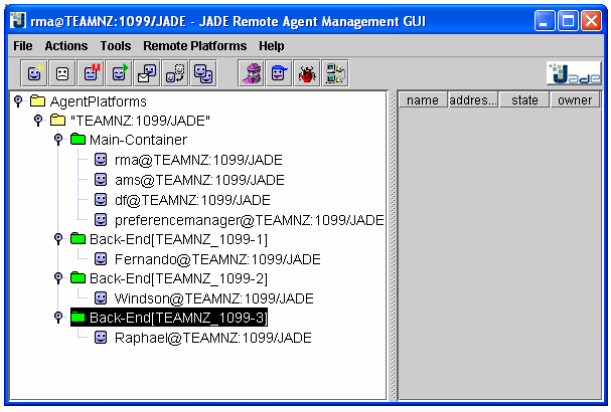


Figure 9.20. Le trois agents créés en utilisant JBuilder s'ajoutent à la liste d'agents de la plate-forme.

[114] En génie logiciel, le *Back-End* est la partie responsable de gérer et traiter les données provenant du *Front-End* (concernant la partie qui interagit avec l'utilisateur)

Lorsque l'utilisatrice « Angela » se connecte à la plate-forme à travers le Pocket PC (cf. Figure 9.21, à gauche, un *agent de DM* s'exécutant), du côté du serveur, la plate-forme en est notifiée et prépare un « *Back-End* » pour cette utilisatrice (cf. Figure 9.21, à droite, un *agent proxy* s'exécutant).

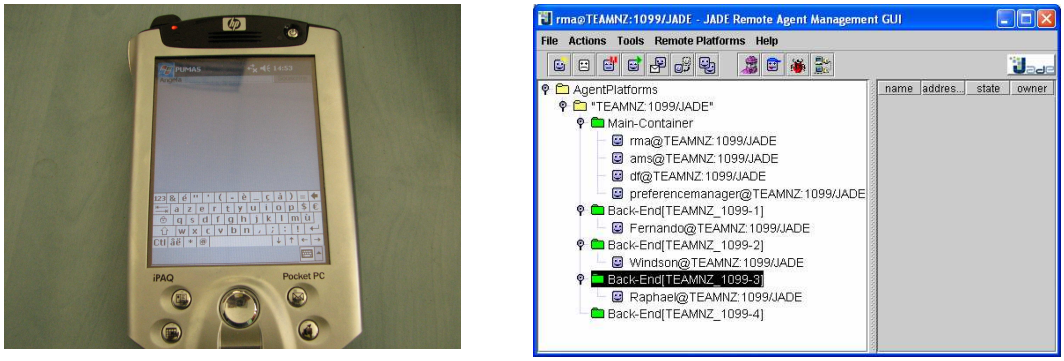


Figure 9.21. L'utilisatrice « Angela » se connecte à PUMAS à travers le Pocket PC (à gauche). A droite, une interface de la plate-forme JADE-LEAP qui montre la création d'un Back-End pour chaque utilisateur qui se connecte à la plate-forme.

L'utilisatrice « *Angela* » donne ensuite son identification qui est envoyée à la plate-forme. La plate-forme crée alors un *agent proxy* qui représente cette utilisatrice (cf. Figure 9.22). Suite à cette création, le nouvel agent apparaît sur la liste de pairs des autres agents connectés. Cette liste est affichée sur l'interface de chaque *agent de DM* associé à chaque *agent de DM* (cf. Figure 9.23).

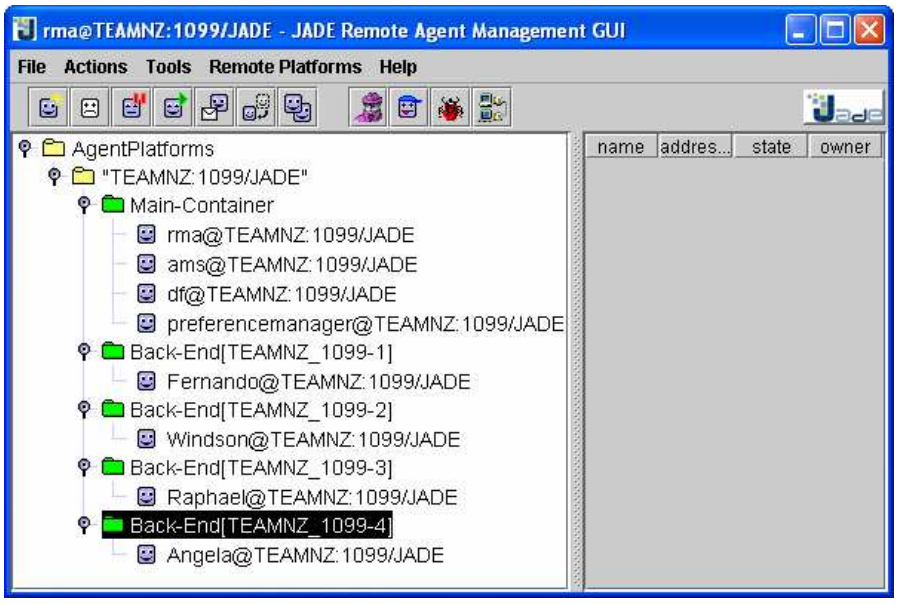


Figure 9.22. Mise à jour de l'interface de la plate-forme JADE-LEAP avec l'agent correspondant à la nouvelle utilisatrice « Angela ».

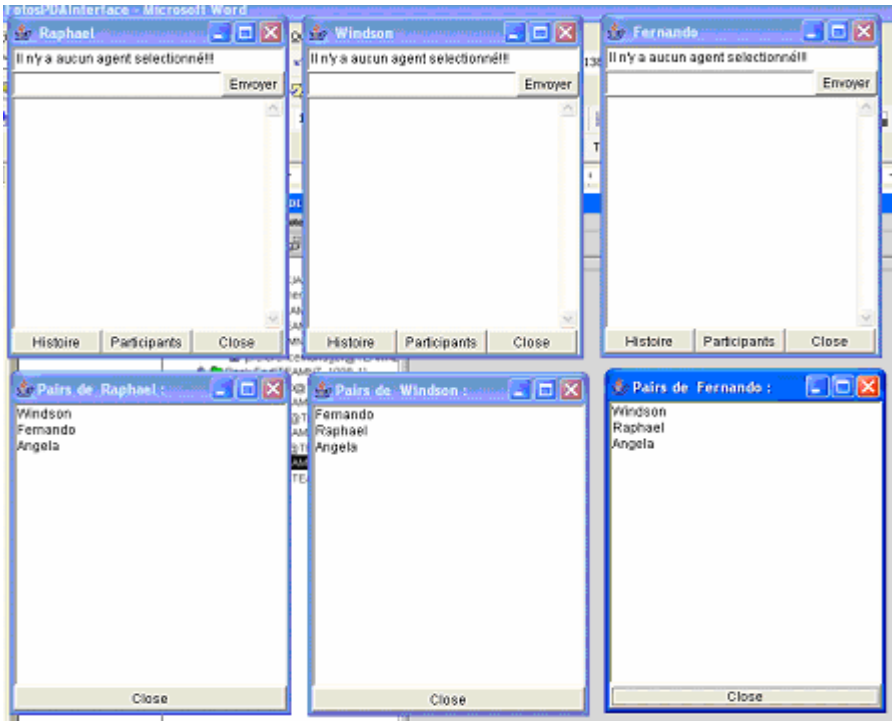


Figure 9.23. Mise à jour des interfaces des utilisateurs simulés en ajoutant l'utilisatrice « Angela » comme pair pour chacun.

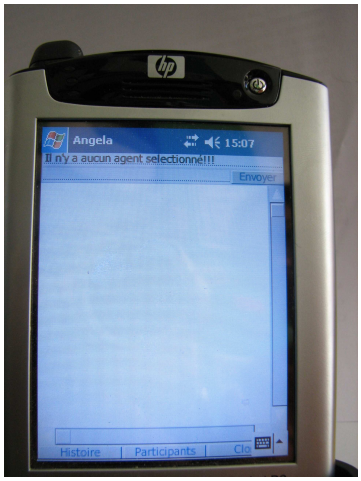


Figure 9.24. Interface de l'utilisateur pour le Pocket PC. L'agent correspondant à l'utilisatrice « Angela » est connecté.

L'interface sur le *DM* de l'utilisateur (cf. Figure 9.24) est la même que celle utilisée par les utilisateurs simulés (cf. Figure 9.19). L'interface permet à l'utilisateur de : *i)* envoyer un message à ses pairs ; *ii)* connaître la liste des participants (les agents connectés, voir le bouton « *Participants* », Figure 9.24) ainsi que la liste de tous les agents qui ont été connectés au système (voir le bouton « *Histoire* », Figure 9.24) et, *iii)* quitter la session (cf. le bouton

« *Close* », Figure 9.24). Pour cette application, les agents déconnectés seront affichés sur l'interface utilisateur avec leur adresse complète.

Chaque *agent proxy* (cf. Figure 9.23) est automatiquement notifié par l'*agent contrôleur de connexions*, de la connexion d'« *Angela* » qui peut aussi être vérifiée sur la plate-forme. L'*agent proxy* (cf. Figure 9.23) envoie cette information à l'*agent de DM* qui ajoute « *Angela* » à la liste de ses agents pairs (agents connectés). L'*agent de DM* correspondant à l'utilisatrice « *Angela* » reçoit aussi de son *agent proxy* la liste de ses agents pairs (cf. Figure 9.25).

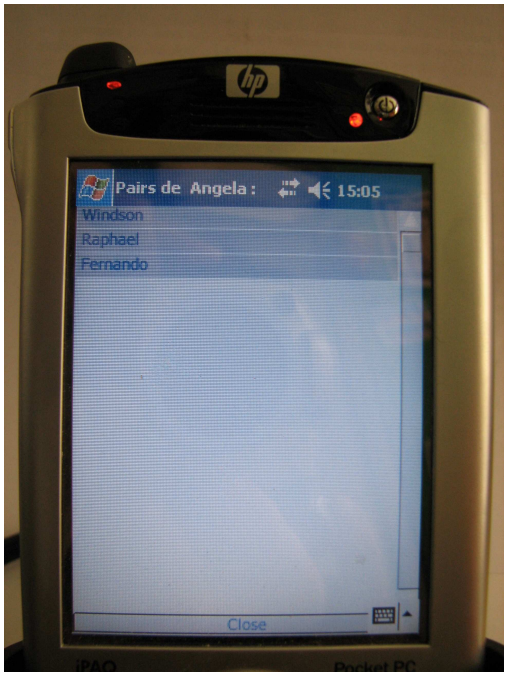


Figure 9.25. Mise à jour de l'interface de l'utilisateur muni d'un Pocket PC. Les trois agents initialement enregistrés sur la plate-forme deviennent des pairs de l'agent correspondant à l'utilisatrice « Angela ».

Suivant le même processus que celui décrit pour la connexion de l'utilisatrice « *Angela* », une nouvelle utilisatrice « *Céline* » peut se connecter en utilisant le même *DM* qu'*Angela*. La plate-forme et ses agents suivent le processus de mise à jour de leur liste de pairs (cf. Figure 9.26).

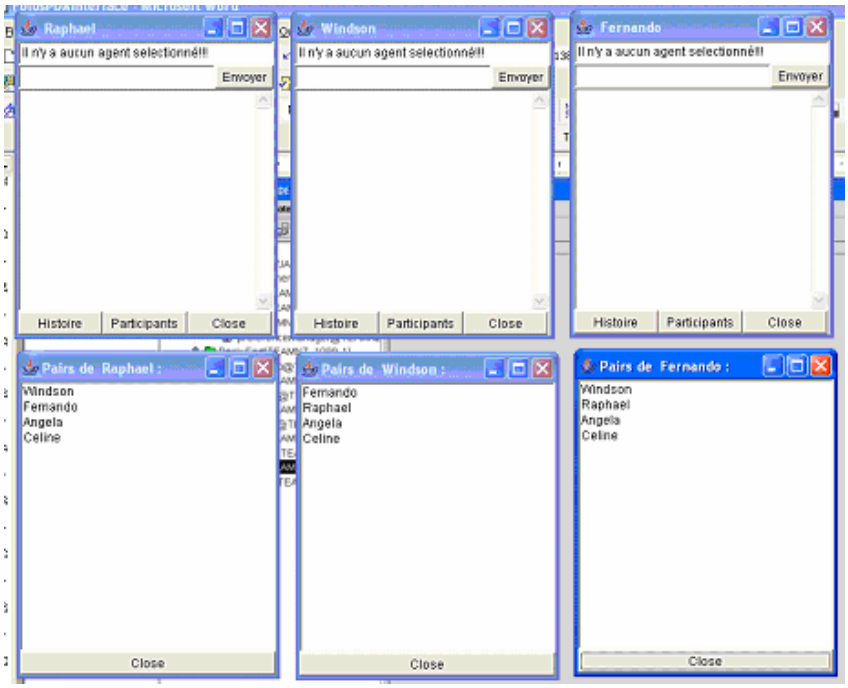


Figure 9.26. Mise à jour des interfaces des utilisateurs simulés en ajoutant l'utilisatrice « Céline » comme pair pour chacun.

L'*agent de DM* (correspondant à l'utilisatrice « *Angela* ») apparaît sur la liste de pairs de l'agent correspondant à l'utilisatrice « *Céline* » (cf. Figure 9.27). Réciproquement, l'*agent de DM*, correspondant à l'utilisatrice « *Céline* », apparaît dans la liste des pairs d'*Angela*.

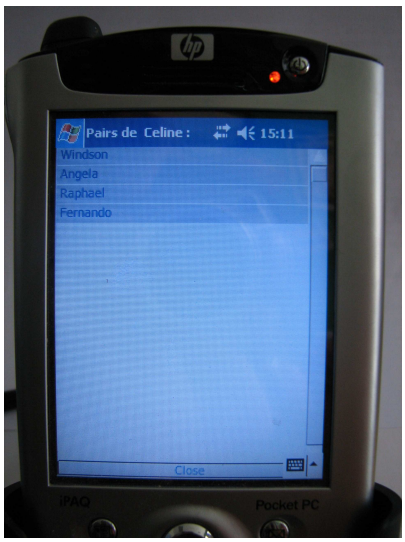


Figure 9.27. Interface de l'utilisatrice « Céline » sur le Pocket PC. Ses agents pairs sont ceux simulés sur le serveur plus celui de « Angela » s'exécutant sur le même Pocket PC.

La Figure 9.28 montre le gestionnaire de tâches du *Pocket PC* présentant les deux instances de l'application d'agents en train d'exécuter : une pour « *Céline* » et une autre pour « *Angela* ».

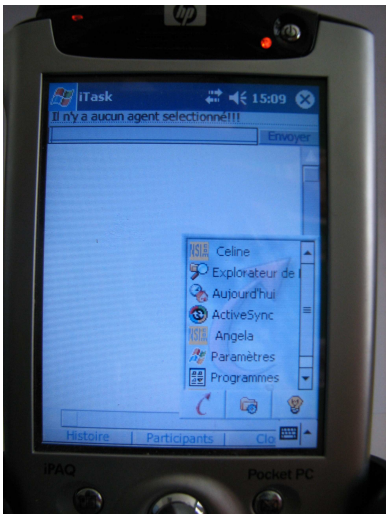


Figure 9.28. Le gestionnaire de tâches du Pocket PC montre les applications d'agents s'exécutant sur le même Pocket PC pour deux utilisateurs différents.

Après la déconnexion de l'utilisateur « *Raphaël* », son *agent proxy* reste encore « *connecté* » sur la plate-forme (cf. Figure 9.29). Cependant, cet agent n'apparaît plus sur la liste de pairs des autres agents montrée sur l'interface de l'*agent de DM* (La Figure 9.30 montre les agents pairs de l'agent « *Angela* »).

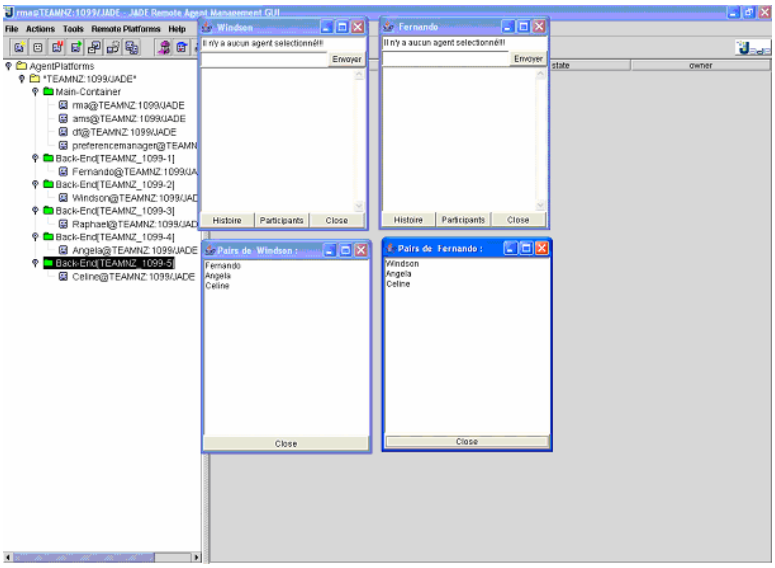


Figure 9.29. Interface de la plate-forme JADE-LEAP. L'agent de DM correspondant à l'utilisateur Raphaël s'est déconnecté. Bien que son agent proxy associé apparaisse comme « connecté » à la plate-forme, il n'apparaît plus sur l'interface des agents de DM.

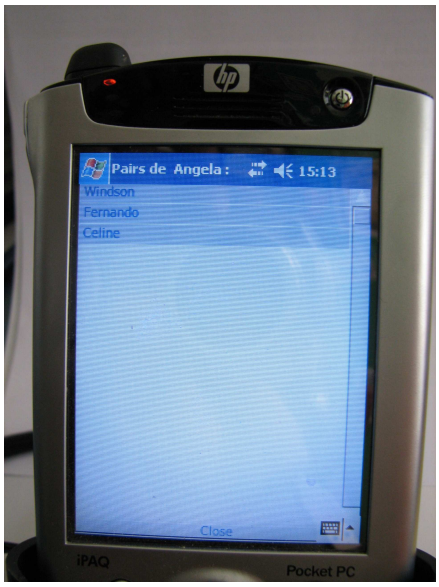


Figure 9.30. Interface de l'utilisateur pour le Pocket PC. L'agent de DM correspondant à l'utilisateur Raphaël s'est déconnecté. Il n'apparaît plus comme pair des agents de DM.

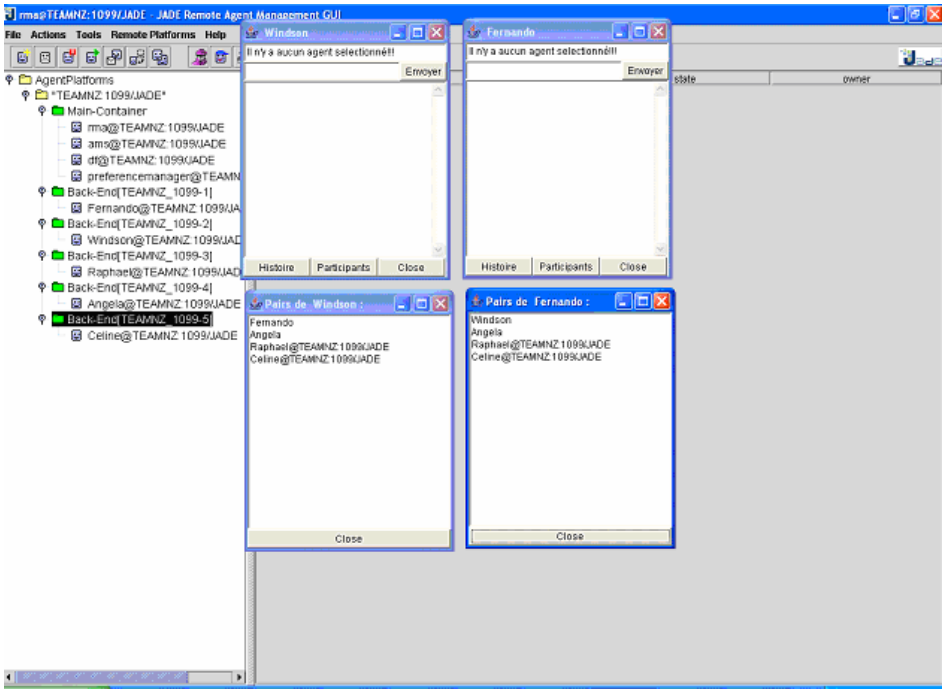


Figure 9.31. Interface de la plate-forme JADE-LEAP. Les agents proxy demandent à l'agent contrôleur de connexions tous les agents proxy qui se sont connectés à la plate-forme (c'est-à-dire, l'historique). L'agent contrôleur de connexions leur envoie une liste dans laquelle les agents déconnectés apparaissent avec leur adresse complète (pour les différencier des agents connectés).

Lorsque les *agents proxy* demandent à l'*agent contrôleur de connexion* l'historique de connexions, ils obtiennent de manière différente la liste des utilisateurs connectés et ceux qui sont déconnectés. Chaque *agent proxy* envoie cette liste à son *agent de DM* associé. La Figure 9.32 montre les utilisateurs *Raphaël* et *Céline* avec l'adresse complète (correspondant à l'adresse complète de leurs agents proxy) car ils se sont déconnectés.

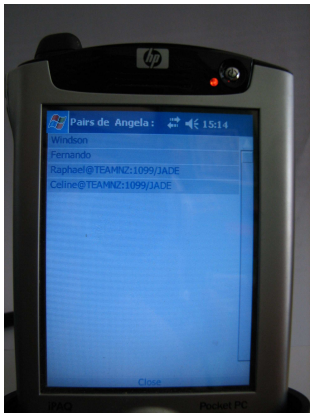


Figure 9.32. Interface de l'utilisateur du Pocket PC. L'agent de DM correspondant à l'utilisatrice « Angela » demande tous les agents qui se sont connectés à la plate-forme. Les agents déconnectés apparaissent avec leur adresse complète.

9.2.2 Envoi de messages entre utilisateurs

Dans la suite, nous allons illustrer l'envoi de messages entre les utilisateurs connectés à *PUMAS* à travers leurs *DM*. Pour les tests, le groupe d'utilisateurs sera composé de *Raphaël*, *Windson*, *Fernando* et *Bogdan* (cf. Figure 9.33) :

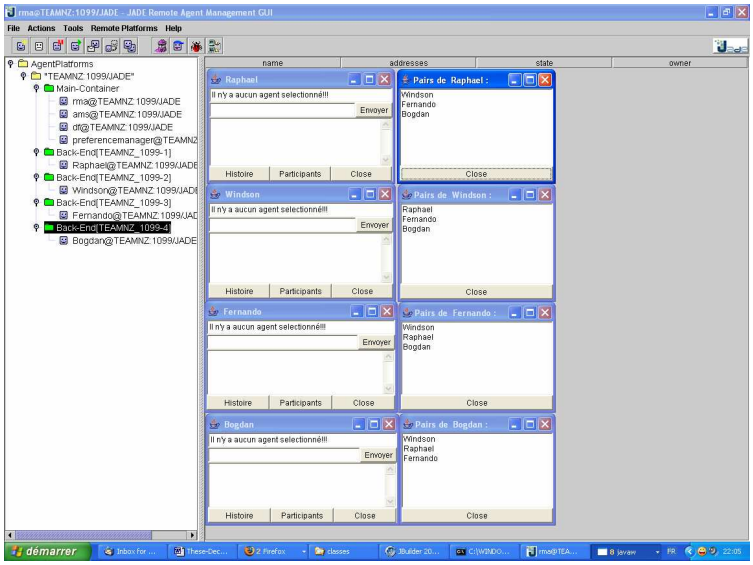


Figure 9.33. Interface de la plate-forme JADE-LEAP et des frames des quatre utilisateurs connectés.

L'utilisateur *Raphaël* peut choisir l'ensemble des destinataires d'un message parmi la liste de ses pairs. Dans cet exemple, *Raphaël* a choisi les utilisateurs *Fernando* et *Bogdan* pour envoyer le message dont le contenu est « *Message pour Fernando et Bogdan* ». La Figure 9.34 montre sur l'interface manipulée par *Raphaël*, dans le champ texte, la chaîne de caractères avec les noms « *Fernando* » et « *Bogdan* ». La Figure 9.35 montre le message reçu uniquement par les agents correspondant aux utilisateurs « *Fernando* » et « *Bogdan* ».

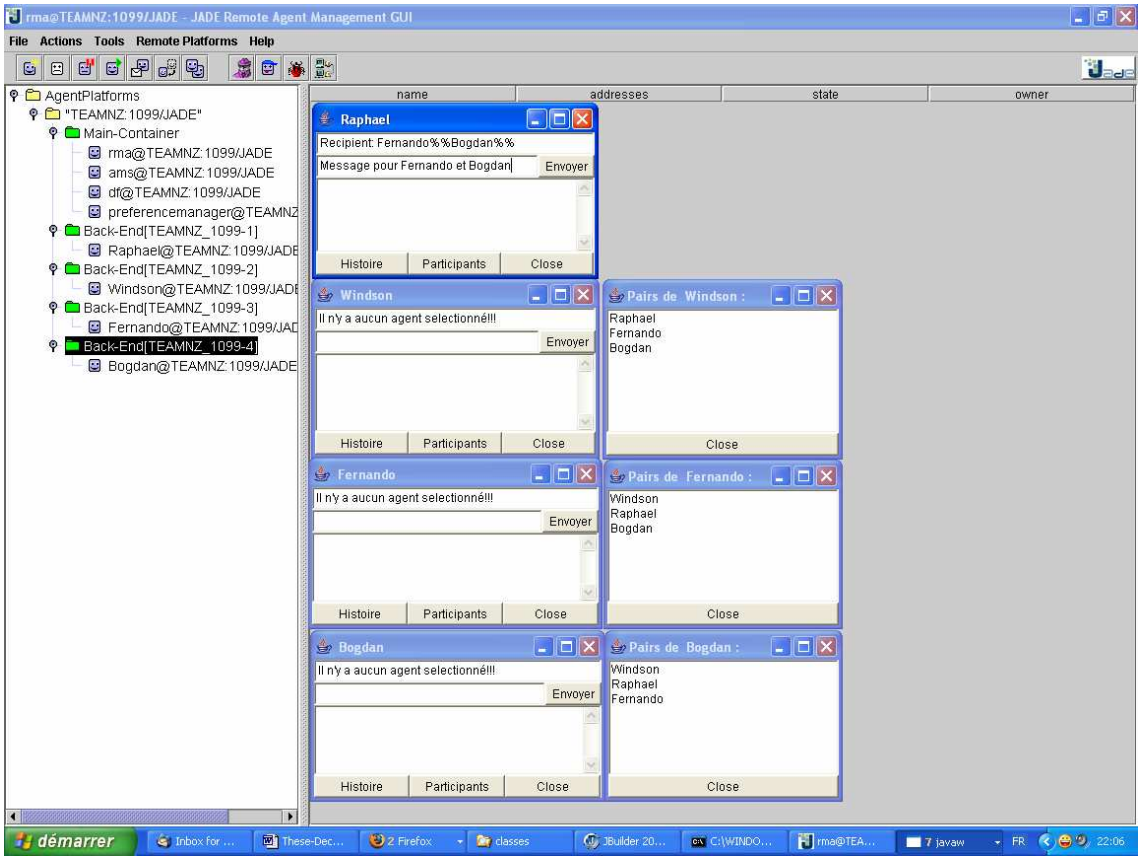


Figure 9.34. Interface de la plate-forme JADE-LEAP. Cette figure montre la plate-forme et les interfaces des agents de DM correspondant à quatre utilisateurs connectés. L'interface de l'agent de DM correspondant à l'utilisateur Raphaël montre que cet utilisateur a sélectionné les utilisateurs Fernando et Bogdan pour leur envoyer le message « Message pour Fernando et Bogdan ».

Si l'utilisateur ne choisit aucun de ses pairs, le message est transmis en *broadcast* : l'*agent de DM* envoie le message à tous les pairs connectés. La Figure 9.35 montre que l'utilisateur Windson n'a sélectionné aucun utilisateur comme destinataire du message « *Message pour tous* !!! ». Ce message a été donc envoyé à tous les utilisateurs connectés et apparaît sur l'interface d'*agent de DM* associé à chaque utilisateur.

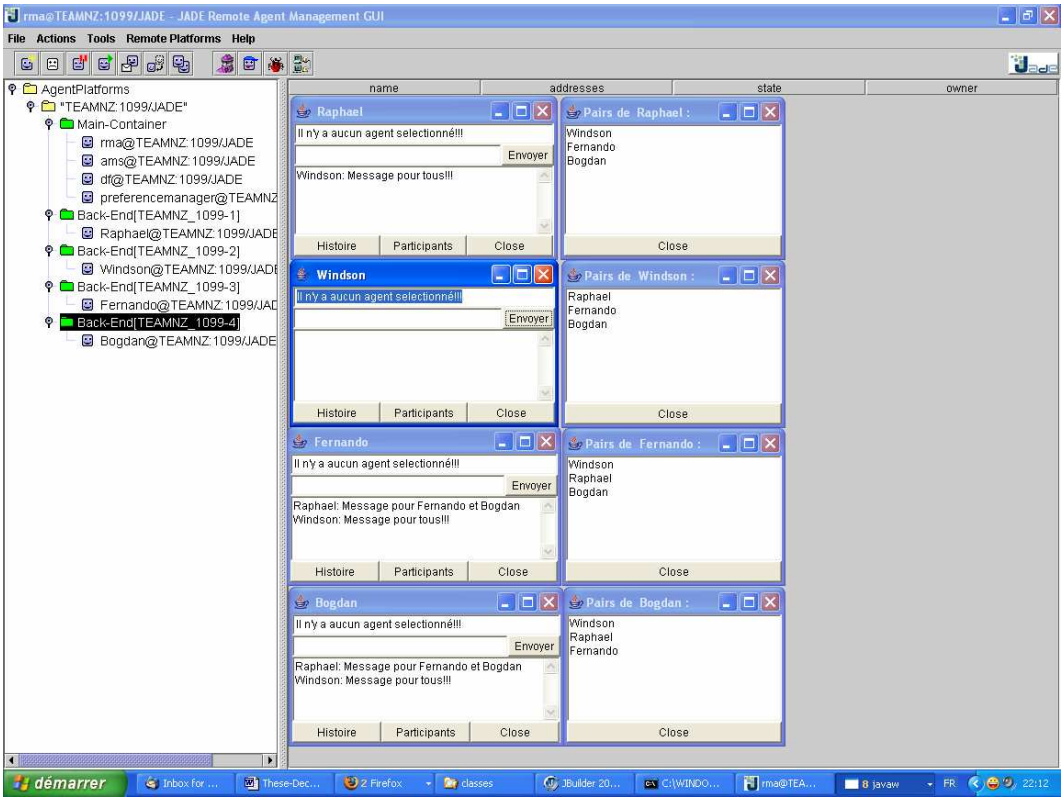


Figure 9.35. Interface de la plate-forme JADE-LEAP. Cette figure montre la plate-forme et l'interface des quatre utilisateurs connectés. L'utilisateur Windson n'a sélectionné aucun utilisateur. Son message a été transmis en broadcast.

9.3. Conclusion

Dans ce chapitre, nous avons présenté des diagrammes *AUML* pour la modélisation des *SMA* de *PUMAS* et pour représenter les interactions entre leurs agents. Puis, nous avons utilisé des diagrammes de classes *UML* pour représenter les trois types de préférences de l'utilisateur et le profil d'un utilisateur nomade pour une session. Nous avons rappelé les caractéristiques les plus importantes des outils que nous utilisons pour la définition des ontologies (*Protégé*) et pour l'implémentation de *PUMAS* (la plate-forme *JADE-LEAP*). Enfin, nous avons introduit des exemples d'interfaces fournies à l'utilisateur pour deux scénarios d'utilisation de *PUMAS* : le premier concerne l'accès et la connexion à *PUMAS* depuis un *DM*. Le deuxième scénario correspond à l'envoi de messages entre des utilisateurs connectés. Dans l'application développée intégrant ces deux scénarios, nous avons mis en évidence un ensemble de problèmes de la plate-forme *JADE-LEAP* par rapport à la connexion, déconnexion et reconnexion des agents, et à l'envoi de messages. Nous avons proposé des solutions aux problèmes concernant la connexion et l'envoi de messages en utilisant *PUMAS*. Nous avons également testé notre application sur des *DM* de type Pocket PC iPAQ 5550.

CONCLUSIONS

10. CONCLUSION ET PERSPECTIVES

10.1. Conclusion

L'ensemble des travaux présentés dans cette thèse étudie l'accès et l'adaptation de l'information pour des *SIW* (*Systèmes d'Informations basé sur le Web*) dans des environnements nomades. Dans ce type d'environnement, l'accès aux *SIW* se fait à travers des *Dispositifs Mobiles* (*DM*) en fournissant à l'utilisateur nomade une information adaptée en considérant son profil ainsi que les caractéristiques et contraintes techniques de son dispositif d'accès. A des fins d'adaptation de l'information, il s'avère nécessaire de disposer de mécanismes permettant de représenter les *préférences de l'utilisateur* ainsi que de générer un *profil contextuel de l'utilisateur* composé de ses préférences pouvant être appliquées en fonction du *contexte d'utilisation* de sa session en cours.

Dans notre proposition, nous pouvons identifier deux contributions principales :

- La première contribution réside dans la conception d'un *framework* basé sur des agents permettant aux utilisateurs nomades d'accéder aux *SIW* et de récupérer l'information la plus appropriée en considérant leurs caractéristiques et celles de leurs dispositifs d'accès ;
- La deuxième contribution concerne la représentation et la gestion des *préférences de l'utilisateur* ainsi que la prise en compte du *contexte d'utilisation* courant de l'utilisateur nomade pour la génération de son profil et l'adaptation de l'information.

Ces contributions sont décrites dans la suite.

Le framework PUMAS pour l'accès et l'adaptation de l'information dans des environnements nomades

Notre première contribution est un *framework* appelé *PUMAS*. Il se base sur des agents et sur une approche *P2P* pour modéliser, concevoir et développer un *Système d'Information Web basé sur des Agents* (*SIWA*). L'architecture de *PUMAS* repose, d'une part, sur les trois niveaux que l'on observe traditionnellement dans les *SIWA* et qui sont représentés par des *SMA* à des fins de *connexion*, de *communication* et de *gestion d'information*. D'autre part, l'architecture de *PUMAS* présente la particularité d'intégrer en plus un *SMA* transversal dédié à l'adaptation (le *SMA d'adaptation*). *PUMAS* fournit aux utilisateurs nomades une information adaptée en tenant compte de leurs caractéristiques, préférences, historique, *etc*. et des contraintes techniques de leurs *DM*.

PUMAS bénéficie également des caractéristiques d'une *architecture P2P* hybride :

- l'indépendance des agents pour se connecter, déconnecter et reconnecter au système ;
- une communication directe entre les pairs (en utilisant l'identification de l'agent et *via* la plate-forme centrale).

PUMAS fournit également un mécanisme pour identifier, authentifier et connaître les pairs d'un agent. *PUMAS* offre par ailleurs des avantages en matière de traitement des requêtes et de recherche d'information. Tout d'abord, *PUMAS* propose un processus de *routage de requêtes* qui analyse les requêtes de l'utilisateur, sélectionne les *Systèmes d'Information* capables d'y répondre et redirige les requêtes vers ces systèmes. Ce processus comporte également une phase de compilation des résultats. Ensuite, il existe une recherche de l'information qui considère les caractéristiques de l'utilisateur et celles de son dispositif d'accès. Cette recherche est *adaptative* puisqu'elle prend en compte le profil de l'utilisateur nomade, les caractéristiques de son *DM*, et en général, les caractéristiques contextuelles de la session en cours. La recherche d'information est également *intelligente* car elle est basée sur la connaissance de l'agent (propre, acquise et inférée) et ses capacités de raisonnement.

En ce qui concerne les connaissances des agents de *PUMAS*, leur représentation est faite à travers quatre ontologies qui décrivent les préférences de l'utilisateur, sa localisation, les caractéristiques de son *DM*, et celles de la session en cours. Afin de gérer et de stocker cette connaissance, nous avons défini des *pièces de connaissances* (nommés « *faits* »), stockées dans des *bases de connaissances* et utilisées par les *agents de PUMAS* pour accomplir leurs tâches. La connaissance gérée et échangée par les *SMA d'information* et *d'adaptation* de *PUMAS* est particulièrement utilisée afin d'adapter l'information et d'accomplir le *processus de routage de requêtes*.

A des fins de modélisation de *PUMAS*, nous avons utilisé les diagrammes *AUML* comme un formalisme pour représenter l'interaction des agents de *PUMAS*. Pour mettre en œuvre *PUMAS*, nous avons choisi *JADE-LEAP* notamment pour son indépendance à l'égard de la plate-forme d'exécution (elle peut s'exécuter sur plusieurs types de *DM* et sur des serveurs). Nous avons testé les exemples qui viennent avec *JADE-LEAP* sur des *DM* (des Pocket PC avec Windows CE, en utilisant crème – *kVM* qui est compatible avec « *Personal Java* », et des *PDA*

avec *PalmOS* en utilisant l'implémentation *MIDP 1.0*). Ces exemples ont montré des problèmes typiques des applications s'exécutant sur des *DM* (par exemple, des problèmes de connexion, de déconnexion, et de reconnexion des utilisateurs, d'envoi de messages, d'interfaces complexes sur des *DM*, *etc*.) qui ont été résolus en utilisant les principes d'accès et d'adaptation de *PUMAS*. L'exécution *stand-alone* de *JADE-LEAP* a montré son instabilité sur les *Pocket PC*, ce qui ne permet pas une implémentation d'un *système P2P pur*. Nous avons pallié cette difficulté en mettant en œuvre une exécution *partagée* (« *Split* ») simulant une *architecture P2P hybride.*

Par rapport aux aspects présentés dans le Tableau 2 qui comparait des architectures basées sur des agents pour l'accès à des *SIW* à travers des *DM* (cf. section 5.1), *PUMAS* les traite de la manière suivante (cf. Tableau 7) :

Aspect	PUMAS	Commentaire
Distribution de données	Serveurs et *DM*	– Architecture *P2P* hybride. – Agents s'exécutent sur : *i)* des *DM, ii)* la plate-forme centrale de *PUMAS*, et *iii)* des systèmes d'information dans lesquels l'information est recherchée. – Recherche de l'information sur des serveurs et des *DM.*
Protocoles de communication	TCP/IP, http	Protocoles pour le Web
Types de données multimédia	Image, vidéo, texte, son	Formats supportés par le dispositif d'accès et sous lesquels l'information est disponible.
Aspects pris en compte pour la définition du profil	Préférences de l'utilisateur et contexte d'utilisation	Proposition d'un mécanisme de génération du profil contextuel de l'utilisateur en sélectionnant les préférences (d'activité, résultat et d'affichage) pouvant être satisfaites par rapport au contexte d'utilisation de la session en cours (contexte composé de la localisation de l'utilisateur, les caractéristiques du *DM*, les droits d'accès de l'utilisateur et ses activités)
Type de DM	PDA, téléphone portable, Pocket PC, Palm, ordinateur portable, *etc*.	Pour chaque *DM*, ses caractéristiques sont décrites dans un fichier écrit en *OWL*, stocké sur le *DM*, et échangé entre des agents de *PUMAS.*
Mécanisme de détection de localisation	Acquisition avec un dispositif *GPS* ou avec des méthodes proposées par Nieto-Carvajal *et al.* [Niet04]	– La spécification de la localisation est décrite dans un fichier écrit en *OWL* stocké sur le *DM*, et échangé entre des agents de *PUMAS*. – La localisation peut être obtenue de différentes manières et sera utilisée afin d'adapter l'information. – La localisation représente une partie du contexte d'utilisation.

Tableau 7. Traitement dans PUMAS des aspects de base des architectures qui modélisent des SMA.

Gestion et représentation des préférences de l'utilisateur

La deuxième contribution de cette thèse permet d'adapter au mieux l'information aux besoins d'un utilisateur nomade accédant à un *SIW* en tenant compte du *contexte d'utilisation courant* (c'est-à-dire, la localisation, les caractéristiques de l'utilisateur et celles de son dispositif d'accès mobile). Nous pouvons citer quatre apports de notre proposition à des fins d'adaptation.

Le premier apport concerne la formalisation de la notion de *préférence de l'utilisateur*. Cette formalisation offre un support à la représentation de trois types de *préférences de l'utilisateur* :

- Les *préférences d'activité* renseignent sur les *activités* qu'un utilisateur planifie d'accomplir dans le système et sur la façon dont elles s'enchaînent (c'est-à-dire de manière séquentielle, concurrente et/ou conditionnelle).
- Les *préférences de résultat* indiquent une organisation attendue des contenus délivrés par ces activités.
- Les *préférences d'affichage* expriment la manière dont l'utilisateur souhaite voir l'information s'afficher sur son *DM*.

Le deuxième apport correspond à l'utilisation des caractéristiques du *contexte d'utilisation* de la session en cours pour un utilisateur nomade afin de construire son profil. Le *contexte d'utilisation* est constitué de l'information sur la localisation de l'utilisateur, les caractéristiques du *DM*, les droits d'accès de l'utilisateur et ses activités.

Le troisième apport comporte un *algorithme de correspondance contextuelle* dont le résultat est un *profil multidimensionnel contextuel* pour un utilisateur spécifique, lors d'une session donnée. Cet algorithme analyse chaque *préférence de l'utilisateur* afin d'évaluer si celle-ci peut être satisfaite par rapport au *contexte d'utilisation*. Ce contexte permet au système de sélectionner seulement les préférences qui sont compatibles avec les activités de l'utilisateur.

Par rapport aux dimensions considérées par les travaux qui représentent un profil multidimensionnel de l'utilisateur (cf. Tableau 4, section 5.2), notre représentation se positionne de la manière suivante (cf. Tableau 8) :

Dimension	**Notre représentation**	**Commentaire**
Donnés personnelles	+	L'information de l'utilisateur est stockée dans un fichier écrit en *OWL*.
Centres d'intérêts	+	Ils font partie du *contexte d'utilisation*.
Historique dans le système	+	– Gestion de *préférences spécifiques* (pour la session en cours), et de *préférences générales* (pour toutes les sessions). – Génération d'un profil basé sur des profils de sessions précédentes s'il n'existe pas de préférences spécifiées pour la session en cours.
Besoins d'information	+	– Ils font partie du contexte d'utilisation. – Les préférences d'activité permettent d'exprimer la manière dont l'utilisateur souhaite exécuter de telles activités.
Préférences de l'utilisateur	+	Représentation et formalisation de trois types de préférences : d'*activité*, de *résultat* et d'*affichage*.
Nouvelle dimension : *contexte d'utilisation*	+	Utilisation de la représentation du contexte d'utilisation de Kirsch-Pinheiro [Kirs06], en prenant les concepts liés à un environnement nomade mais en ignorant les concepts liés au travail coopératif

Tableau 8. Dimensions considérées dans notre représentation du profil de l'utilisateur.

Le quatrième apport repose sur la *gestion de conflits* entre les *préférences de l'utilisateur*. Ces conflits ne sont pas gérés par la version actuelle de l'*algorithme de correspondance*

contextuelle. Cependant, nous avons identifié certains conflits-types, leurs causes et la manière dont le système doit réagir afin de les résoudre, ou du moins afin d'informer l'utilisateur de leur existence. Parmi les causes de conflits que nous avons identifiées, deux sont mises en évidence :

- L'incompatibilité entre les préférences et les droits d'accès de l'utilisateur dans le *Système d'Information* ;
- L'incompatibilité entre les *préférences de l'utilisateur* et les contraintes techniques de son *DM*.

En résumé, par rapport aux travaux basés sur des agents qui adaptent l'information dans des environnements nomades (cf. Tableau 5, section 5.2), notre proposition se positionne de la manière suivante (cf. Tableau 9) :

Aspect	**Notre proposition**	**Commentaire**
Profil de l'utilisateur	+	Généré par l'*algorithme de correspondance contextuelle* qui considère les *préférences de l'utilisateur* pouvant être satisfaites par rapport au *contexte d'utilisation* de la session en cours.
Localisation de l'utilisateur	+	– Elle fait partie du *contexte d'utilisation*. – Acquise avec un dispositif GPS, fournie par l'utilisateur ou obtenue en utilisant une des méthodes proposées par Nieto-Carvajal *et al.* [Niet04]
Préférences de l'utilisateur	+	– Classifiés en trois types : d'activité, de résultat et d'affichage. – Utilisées par le *filtre de contenu* qui adapte l'information par rapport aux caractéristiques de l'utilisateur. Ce filtre considère particulièrement les *préférences d'activité* et *de résultat*.
Adaptation aux différents types de média	+	A travers un *filtre d'affichage* considérant les caractéristiques du *DM* et les *préférences d'affichage* de l'utilisateur
Relations entre les agents	+	Basées sur l'organisation des agents dans quatre *SMA* en utilisant une architecture *P2P hybride*.
Mécanismes de routages de requêtes	+	Réduits à un mécanisme de routage proposé (accompli par l'*agent de routage* appartenant au *SMA d'Information*) et composé de trois activités : – analyse de la requête, – sélection des systèmes d'information capables d'y répondre et, – redirection de la requête vers les systèmes d'information sélectionnés.
Distribution de l'information entre DM	+	– Prise en compte par l'*agent de routage* qui peut sélectionner des systèmes d'information s'exécutant sur des *DM*. – Supportée par les *agents de DM* qui gèrent l'information du *DM* et qui peuvent jouer le rôle d'*agents de systèmes d'information*.

Tableau 9. Aspects pris en compte par notre proposition basée sur des agents pour adapter l'information à l'utilisateur.

10.2. Perspectives

Nos contributions à l'accès aux *SIW* à travers des *DM* et à l'adaptation de l'information dans des environnements nomades ouvrent des perspectives sur plusieurs aspects. Nous présentons tout d'abord les perspectives concernant le framework *PUMAS* puis celles liées à notre formalisation, représentation et gestion des *préférences de l'utilisateur* ainsi qu'à notre définition du *profil contextuel de l'utilisateur*.

Perspectives pour le framework PUMAS

La première perspective concerne la définition des algorithmes et des stratégies pour la mise en œuvre des *filtres de contenu* et *d'affichage*. Nous envisageons l'utilisation du *Modèle d'Access Progressif* (*MAP*) [Vill02] pour définir le *filtre de contenu*, et le *modèle hypermédia* [Vill02] pour définir le *filtre d'affichage*. Nous envisageons aussi d'utiliser les règles de correspondance de transformations présentées par [Berh06] qui déterminent les transformations de média requises pour adapter des contenus multimédias dans des environnements pervasifs en considérant les capacités du dispositif d'accès, les préférences de l'utilisateur, les conditions du réseau, *etc*.

La deuxième perspective consiste en l'exploitation des aspects spatio-temporels pour les communications des agents. Cette exploitation requiert l'extension des langages de communication des agents (*ACL*) avec un ensemble de nouveaux *actes de communication* tels que « *query-when* », « *query-where* », « *query-close* », qui permet aux agents de demander de l'information à d'autres agents à un moment et à un endroit donné.

Nous avons déjà défini l'algorithme pour l'*analyse d'une requête* qui correspond à la première activité du processus de *routage de requêtes*. La troisième perspective de notre travail concerne la définition des algorithmes pour les deux autres activités de ce processus :

- la sélection des systèmes d'information capables de répondre aux requêtes de l'utilisateur ;
- la redirection des requêtes.

Nous envisageons aussi de définir l'algorithme pour la compilation et l'analyse des résultats des requêtes provenant d'un ou plusieurs *systèmes d'information* (s'exécutant sur des *DM* ou des serveurs).

La quatrième perspective consiste à compléter l'implémentation des composants de *PUMAS*. Le travail d'implémentation sera accompagné de tests (y compris l'accès en utilisant des *DM*) pour aboutir à un *framework* stable.

Perspectives pour la gestion et représentation de préférences de l'utilisateur

La première perspective concerne l'amélioration de l'algorithme de *correspondance contextuelle* qui génère le *profil contextuel de l'utilisateur*. Cette amélioration concerne la mise en œuvre de deux mécanismes. Le premier permet d'évaluer et d'établir les priorités entre les préférences (nous avons défini dans notre proposition que les *préférences spécifiques* ont une

priorité plus haute que les *préférences générales*). Le second permet de détecter et résoudre des éventuels conflits qui pourraient survenir au moment d'ajouter une préférence au *profil contextuel* (nous avons défini dans notre proposition des conflits-types qui peuvent survenir entre la préférence à ajouter et celles appartenant au *profil contextuel*, ainsi que la manière dont le système doit réagir lors de la présentation d'un de tels conflits).

La deuxième perspective correspond à la *représentation* et à la *résolution* de conflits entre les *préférences de l'utilisateur*.

Pour la *représentation*, nous proposons utiliser le langage présenté par Bell [Bell01] permettant la représentation d'événements contradictoires simultanés. Cette représentation indique que lorsque deux événements contradictoires surviennent (dans notre cas, l'ajout d'une préférence peut produire des conflits avec d'autres préférences appartenant au profil), il est possible de définir que les effets produits par un événement puissent annuler les effets produits par un autre événement. Par exemple, le fait de prendre en compte certaines caractéristiques du *contexte d'utilisation* peut permettre de décider si une préférence sera incluse ou non dans le profil.

Pour la *résolution* de conflits, nous envisageons d'utiliser des méthodes telles que :

- L'*argumentation* [Amgo02] qui établit des arguments afin d'une part, de définir des relations (par exemple hiérarchie, priorités dans notre proposition) entre les *préférences de l'utilisateur* et, d'autre part, déterminer quelles préférences seront considérées dans un scénario spécifique (par exemple, sélectionner les préférences de l'utilisateur qui peuvent être incluses dans le *profil contextuel de l'utilisateur* pour une session spécifique).
- Les règles de partitionnement (« *Branching rules* ») présentées par Herbstritt *et al.* [Herb03] associent une valeur à une préférence indiquant sa probabilité d'être sélectionnée ou appliquée.
- Des protocoles d'Interaction (tels que « *Contract Net Protocol* » [Smit83] ou « *Partial Global Planning* » [Brus96] [Buss02] [Durf91]) qui établissent un ensemble d'actions à accomplir si une situation spécifique (un conflit dans notre approche) survient. Dans notre proposition, les actions sont les mécanismes dont le système dispose pour réagir à la présence d'un conflit.

Le travail consistera à étudier chacune de ces approches, à les comparer et à évaluer leurs apports pour la résolution des conflits dans un *Système Multi-Agents* en charge d'adapter l'information. Nous souhaitons tout d'abord établir une classification des applications-types permettant d'identifier quelle(s) approche(s) semble(nt) la (les) plus adéquate(s) pour chaque classe d'applications. Les classes d'applications seront par exemple construites en considérant la structure du groupe d'agents, la nature des tâches à accomplir, les caractéristiques spatio-temporelles de l'application (par exemple, réponses en temps réel attendues), *etc.* Si la classification nous permet d'identifier des applications types partagées par ces approches, nous chercherons à identifier si le fait de les combiner présente des avantages significatifs pour la résolution des conflits.

11. REFERENCES

[Agos00] Agoston, T., Ueda, T., Nishimura, Y. Pervasive Computing in a Networked World. In: CDProceedings of the 10th Annual Internet Society Conference (INET 2000) (Yokohama, Japon, July 18-21, 2000) [En ligne]. Disponible sur : http://www.isoc.org/inet2000/cdproceedings/3a/3a_1.htm#s1. (Consulté en Septembre 2006).

[Agos04] Agostini, A., Moro, G. Identification of Communities of Peers by Trust and Reputation. In: Bussler, C., Fensel, D. (eds.): Proceedings of the 11th International Conference in Artificial Intelligence: Methodology, Systèmes, and Applications (AIMSA 2004) (Varna, Bulgaria, September 2-4, 2004), Lecture Notes in Computer Science, vol. 3192, Springer-Verlag, Berlin Heidelberg (2004), pp. 85-95.

[Alba05] Albayrak, S., Wollny, S., Varone, N., Lommatzsch, A., Milosevic D. Agent Technology for Personalized Information Filtering: The PIA-System. In: Liebrock, L. (eds.): Proceedings of the 20th Annual ACM Symposium on Applied (SAC 2005) (Santa Fe, USA, March 13-17, 2005), ACM Press, New York, NY (2005), pp. 54-59.

[Amgo02] Amgoud, L., Parson, S. An Argumentation Framework for Merging Conflicting Knowledge Bases. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.): Proceedings of the European Conference on Logics in Artificial Intelligence (JELIA 2002) (Cosenza, Italy, September, 23-26, 2002), Lecture Notes in Computer Science, vol. 2424, Springer-Verlag, Berlin Heidelberg (2002), pp. 27-205.

[Baad03] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. The Description Logic Handbook. Theory, Implementation and Applications. Cambridge University Press (2003), 574 pages.

[Bail02a] Bailey, C., El-Beltagy, S.R., Hall, W. Link Augmentation: A Context-Based Approach to Support Adaptive Hypermedia. In: Reich, S., Tzagarakis, M., De Bra, P. (eds.): Proceedings of the International Workshops on Hypermedia: Openness, Structural Awareness, and Adaptivity (OHS-7, SC-3, and AH-3) (Aarhus, Denmark,

August 14-18, 2001), Lecture Notes in Computer Science, vol. 2266, Springer-Verlag, Berlin Heidelberg (2002), pp. 239-251.

[Bail02b] Bailey, C., Hall, W., Millard, D.E., Weal, M.J. Towards Open Adaptive Hypermedia. In: De Bra, P., Brusilovsky P., Conejo, R. (eds.): Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002) (Malaga, Spain, May 29-31, 2002), Lecture Notes in Computer Science, vol. 2347, Springer Verlag, Berlin Heidelberg (2002), pp. 36-46.

[Bass95] Bassiliades, N., Gray, P.M.D. CoLan: A functional constraint language and its implementation. In: Data & Knowledge Engineering, vol 14, no 3 (February 1995), pp. 203-249.

[Bell01] Bell, J. Simultaneous Events: Conflicts and Preferences. In: Benferhat, S., Besnard, P. (eds.): Proceedings of the 6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2001) (Toulouse, France, September 19-21, 2001), Lecture Notes in Computer Science, vol. 2143, Springer-Verlag, Berlin Heidelberg (2001), pp. 714-725.

[Bell06] Bellifemine, F., Caire, G., Trucco, T., Rimassa G. JADE Programmer's Guide [En ligne]. http://jade.cselt.it/ (Consulté en Septembre 2006).

[Belo04] Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M.C., Palinginis, A. Interplay of Content and Context. In: Koch, N., Fraternali, P., Wirsing, M. (eds.): Proceedings of the 4th International Conference on Web Engineering (ICWE 2004) (Munich, Germany, July 26-30, 2004), Lecture Notes in Computer Science, vol. 3140, Springer-Verlag, Berlin Heidelberg (2004), pp. 187-200.

[Berh04] Berhe, G., Brunie, L., Pierson, J.M. Modeling Service-Based Multimedia Content Adaptation in Pervasive Computing. In: Proceedings of the 1st Conference on Computing Frontiers (CF 2004) (Ischia, Italy, April 14 - 16, 2004), ACM Press, New York, NY (2004), pp. 60-69.

[Berh06] Berhe, G. Accès et adaptation de contenus multimédia pour les systèmes pervasifs. Thèse de doctorat, Institut National des Sciences Appliquées de Lyon, Lyon, 25 Septembre 2006.

[Beus00] Beuster G., Thomas B., Wolff C. Ubiquitous Web Information Agents. In: Proceedings of the Workshop on Artificial Intelligence in Mobile Systems and European Conference on Artificial Intelligence (ECAI'2000) (Berlin, Germany, August 20-25, 2000) [En ligne]. Disponible sur : http://www.uni-koblenz.de/~gb/articles/aims2000/article.pdf. (Consulté en Janvier 2004).

[Biru05] Birukov, A., Blanzieri E., Giorgini, P. Implicit: An Agent-Based Recommendation System for Web Search. In: Aarts, H., Westra, J. (eds.): Proceedings of the 4th International Conference on Autonomous Agent and Multi-Agent Systems (AAMAS 2005) (Utrecht, Netherlands, July 25-29, 2005), ACM Press, New York, NY (2005), pp. 618-624.

[Boni06] Bonifacio, M. A Peer-to-Peer Solution for Distributed Knowledge Management. In: Staab, S., Stuckenschmidt (eds.): Semantic Web and Peer-to-Peer. Springer-Verlag Berlin Heidelberg (2006), pp. 323-334.

[Bouc06] Bouchard, H., Nie, J.Y. Modèles de langues appliquées à la recherche d'information contextuelle. Actes de CORIA 2006 (Lyon, France, 15-17 mars 2006), pp. 213-224.

[Bouz05] Bouzeghoub, M., Kostadinov, D. Personnalisation de l'information : aperçu de l'état de l'art et définition d'un modèle flexible de profils. Actes de CORIA 2005 (Grenoble, France, 9-11 mars, 2005), pp. 201-218.

[Brun03] Brunkhorst, I., Dhraief, H., Kemper, A., Nedjl W., Wiesner, C. Distributed Queries and Query Optimization in Schema-Based P2P Systems. In: Aberer, K., Kalogeraki, V., Koubarakis, M. (eds.): Proceedings of the 1st International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P) (Berlin, Germany September 7-8, 2003), Lecture Notes in Computer Science, vol. 2944, Springer-Verlag, Berlin Heidelberg (2003), pp. 184-199.

[Brus96] Brusilovsky, P. Methods and Techniques of Adaptive Hypermedia. In: User Modeling and User Adapted Interaction, vol 6, no 2-3 (1996), pp. 87-129.

[Brus00] Brusilovsky, P. Adaptive Hypermedia: From Intelligent Tutoring Systems to Web Based Education. In: Gauthier, G., Frasson, C., VanLehn, K. (eds.): Proceedings of the International Conference on Intelligent Tutoring Systems (ITS 2000) (Montreal, Canada, June 19-23, 2000), Lecture Notes in Computer Science, vol. 1839, Springer-Verlag, Berlin Heidelberg (2000), pp. 1-7.

[Bucc05] Buccella, A., Cechich, A., Brisaboa, N.R. Ontology-Based Data Integration Methods: A Framework for Comparison. In: Arenas, a., Perez, J., Carrillo, E. (eds.): Colombian Journal of Computation. vol. 6, no 2. Décembre 2005 [En ligne]. Disponible sur: http://www.unab.edu.co/editorialunab/revistas/rcc/pdfs/r61_art3_c.pdf (Consulté en Septembre 2006).

[Bucu06] Bucur, O., Beaune, P., Boissier, O. Steps Towards making Contextualized Decisions: how to do what you can, with what you have, where you are. In: Roth-Berghofer, T., Schulz, S., Leake, D.B. (eds.): Proceedings of the 2nd International Workshop on Modeling and Retrieval of Context (MRC 2005) (Edinburgh, UK, July 31 - August 1, 2005) Revised Selected Papers, Lecture Notes in Computer Science, vol. 3946, Springer-Verlag, Berlin, Heidelberg (2006), pp. 62-85.

[Buss02] Bussmann, S., Jennings, N., Wooldridge, M. Re-use of Interaction Protocols for Agent-Based Control Applications. In: Giunchiglia, F., Odell, J., Weib, G. (eds.): Proceedings of the 3rd International Workshop (AOSE 2002) (Bologna, Italy, July 15, 2002), Revised Papers, Lecture Notes in Computer Science, vol. 2585, Springer-Verlag, Berlin Heidelberg (2002), pp. 73-87.

[CaiF04] Cai, M., Frank, M. RDFPeers : A Scalable Distributed RDF Repository based on a Structured Peer-to-Peer Network. In: Feldman, S.I., Uretsky, M., Najork, M., Wills, C.E. (eds.): Proceedings of the 13th International Conference on World Wide Web (WWW 2004) (New York, NY, USA, May 17-20, 2004), ACM 2004 (2004), pp. 650-657.

[Cali04] Calisti, M., Lozza, T., Greenwood, D. An Agent-Based Middleware for Adaptive Roaming in Wireless Network. In: Proceedings of Workshop on Agents for Ubiquitous Computing (UbiAgents04) (Columbia University, New York City, USA July 20, 2004) in conjunction with AAMAS2004 [En ligne]. Disponible sur: http://www.ift.ulaval.ca/~mellouli/ubiagents04/ (Consulté en Septembre 2006).

[Cann01] Cannataro, M., Cuzzocrea, A., Pugliese, A. A Multidimensional Approach for Modelling and Supporting Adaptive Hypermedia Systems. In: Bauknecht, K., Madria,

S.K., Pernul, G. (eds.): Proceedings of the 2nd International Conference on Electronic Commerce and Web Technologies (EC-Web 2001) (Munich, Germany, September 4-6, 2001), Lecture Notes in Computer Science, vol. 2115, Springer Verlag, Berlin Heidelberg (2001), pp. 132-141.

[CaoC04] Cao, J., Chan, K.M., Shea, G.Y.,Guo, M. Location-Aware Information Retrieval for Mobile Computing. In: Yang, L., Guo, M., Gao, G.R., Jha, N.K. (eds.): Proceedings of the Embedded and Ubiquitous Computing (EUC 2004) (Aizu-Wakamatsu City, Japan, August 25-27, 2004), Lecture Notes in Computer Science, vol. 3207, Springer-Verlag, Berlin Heidelberg (2004), pp. 450-459.

[Cara03] Carabelea, C., Boissier, O., Ramparany, F. Benefits and Requirements of Using Multi-agent Systems on Smart Devices. In: Kosch, H., Böszörményi, L., Hellwagner, H. (eds.): Proceedings of the 9th International Euro-Par Conference on Parallel Processing (Euro-Par 2003) (Klagenfurt, Austria, August 26-29, 2003), Lecture Notes in Computer Science, vol. 2790, Springer-Verlag, Berlin Heidelberg (2003), pp. 1091-1098.

[Carr04] Carrillo-Ramos, A., Gensel, J., Villanova-Oliver, M., Martin, H. Modelling with Ubiquitous Agents a Web-based Information System Accessed through Mobile Devices. In: Meersman, R., Tari, Z. (eds.): Proceedings of the Cooperative Information Systems (CoopIS'04). (Larnaca, Cyprus, October 25-29, 2004), Lecture Notes in Computer Science, vol. 3290, Springer-Verlag, Berlin Heidelberg (2004), pp. 264-282.

[Carr05a] Carrillo Ramos, A., Gensel, J., Villanova-Oliver, M., Martin, H. PUMAS: a Framework based on Ubiquitous Agents for Accessing Web Information Systems through Mobile Devices. In: Liebrock, L. (eds.): Proceedings of the 20th Annual ACM Symposium on Applied Computing (SAC2005) (Santa Fe, USA, March 13 -17, 2005), ACM Press, New York, NY (2005), pp. 1003-1008.

[Carr05b] Carrillo-Ramos, A., Gensel, J., Villanova-Oliver, M., Martin, H. A Peer Ubiquitous Multi-agent Framework for Providing Nomadic Users with Adapted Information. In: Despotovic, Z., Joseph, S. and Sartori C. (eds): Post-proceedings of the 4th International Workshop on Agents et P2P Computing (AP2PC 2005) (Utrecht, Netherlands, July 26, 2005), Revised Papers, Lecture Notes in Artificial Intelligence, vol. 4118, Springer-Verlag, Berlin Heidelberg (2006), pp. 159-172.

[Carr06] Carrillo Ramos, A., Villanova-Oliver, M., Gensel, J., Martin, H. Knowledge Management for Adapted Information Retrieval in Ubiquitous Environments. In: Proceedings of the 2nd International Conférence on Web Information Systems and Technologies (WEBIST 2006) (Setubal, Portugal, April 11-13, 2005), Insticc Press, Portugal (2006), pp. 21-29.

[Cole05] Cole, J.J., Gray, M.J., Lloyd, J.W., Ng, K.S. Personalisation for User Agents. In: Aarts, H., Westra, J. (eds.): Proceedings of the 4th International Conference on Autonomous Agent and Multi-Agent Systems (AAMAS 2005) (Utrecht, Netherlands, July 25-29, 2005), ACM Press, New York (2005), pp. 603-610.

[Coll03] Collier, R.W., O'Hare G.M.P., Lowen, T., Rooney, C.F.B. Beyond Prototyping in the Factory of the Agents. In: Marík, V. Müller, J.P., Pechoucek, M. (eds.): Proceedings of the 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003) (Prague, Czech Republic, June 16-18, 2003), Lecture

Notes in Computer Science, vol. 2691, Springer-Verlag, Berlin Heidelberg (2003), pp. 383-393.

[Conl02] Conlan, O., Wade, V., Bruen, C., Gargan, M. Multi-model, Metadata Driven Approach to Adaptive Hypermedia Services for Personalized eLearning. In: De Bra, P., Brusilovsky P., Conejo, R. (eds.): Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002) (Malaga, Spain, May 29-31, 2002), Lecture Notes in Computer Science, vol. 2347, Springer Verlag, Berlin Heidelberg (2002), pp. 100-111.

[Dale00] Dale, J. April Agent Platform Reference Manual [En ligne]. Fujitsu Laboratories of America. 2000. Disponible sur : http://www.nar.fujitsulabs.com/documents/aap-manual.pdf. (Consulté en Février 2004).

[DeCa03] De Carolis, B., Pizzutilo, S., Palmisano, I. D-ME: Personal Interaction en MAS Environnements. In: Brusilovsky, P., Corbett, A.T., de Rosis, F. (eds.): Proceedings of the 9th International Conference of User Modeling (UM 2003) (Johnstown, PA, USA, June 22-26, 2003), Lecture Notes in Computer Science, vol. 2702. Springer-Verlag, Berlin Heidelberg New York (2003), pp. 388-392.

[DeyA99] Dey, A.N., Abowd, G.D. Towards a Better Understanding of Context and Context-Awareness. In: Gellersen, H.W. (eds): Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC'99) (Karlsruhe, Germany, September 27-29, 1999), Lecture Notes in Computer Science, vol. 1707, Springer-Verlag, Berlin-Heidelberg (1999), pp. 304-307.

[Dolo02] Dolog, P., Bieliková, M. Navigation Modelling in Adaptive Hypermedia. In: De Bra, P., Brusilovsky P., Conejo, R. (eds.): Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002) (Malaga, Spain, May 29-31, 2002), Lecture Notes in Computer Science, vol. 2347, Springer Verlag, Berlin Heidelberg (2002), pp. 586-591.

[Dosh05] Doshi, P., Gmytrasiewics, P. Approximating State Estimation in Multiagent Settings Using Particle Filters. In: Aarts, H., Westra, J. (eds.): Proceedings of the 4th International Conference on Autonomous Agent and Multi-Agent Systems (AAMAS 2005) (Utrecht, Netherlands, July 25-29, 2005), ACM Press, New York, NY (2005), pp. 320-327.

[Durf91] Durfee, E.H., Lesser, V.R. Partial global planning: a coordination framework for distributed hypothesis formation. In: IEEE Transactions on systems, man and cybernetics, vol. 21, no. 5 (1991), pp. 1167-1183.

[ElFa04] El Fallah-Seghrouchni, A., Suna, A. CLAIM: A Computational Language for Autonomous, Intelligent and Mobile Agent. In: Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.): Proceedings of the Programming MAS (PROMAS 2003) (Melbourne, Australia, July 15, 2003), Lecture Notes in Artificial Intelligence, vol. 3067, Springer-Verlag, Berlin Heidelberg (2004), pp. 90-110.

[Fini94] Finin, T., Fritzson, R., McKay, D., McEntire, R. KQML as an Agent Communication Language. In: Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94) (Gaithersburg, Maryland, November 29 - December 2, 1994), ACM Press, New York, NY (1994), pp. 456-463.

[Freu03] Freuder, E.C., Likitvivatanavong, C., Moretti, M. Rossi, F., Wallace, R.J. Computing Explanations and Implications in Preference-Based Configurators. In: O'Sullivan, B. (ed.): Proceedings of the Recent Advances in Constraints, Joint ERCIM/CologNet International Workshop on Constraint Solving and Constraint Logic Programming (Cork, Ireland, June 19-21, 2002), Selected Papers, Lecture Notes in Computer Science, vol. 2627, Springer-Verlag, Berlin Heidelberg (2003), pp. 76-92.

[Frie06] Friedman-Hill, E. Jess 7.0 manual [En ligne] Sandia National Laboratories. October 2006. DRAFT. Disponible sur: http://www.jessrules.com/jess/docs/70/ (Consulté en Octobre 2006).

[Gand04] Gandon, F., Sadeh, N. Semantic Web Technologies to Reconcile Privacy and Context Awareness. Journal of Web Semantics [En ligne]. vol. 1, no. 3. (October 31, 2004). Disponible sur : http://www.websemanticsjournal.org/ps/pub/2004-17 (Consulté en Septembre 2006).

[Garl03] Garlatti, S., Iksal, S. Declarative Specifications for Adaptive Hypermedia Based on a Semantic Web Approach. In: Brusilovsky, P., Corbett, A., de Rosis, F. (eds.): Proceedings of the 9th International Conference on User Modeling 2003 (UM 2003) (Johnstown, PA, USA, June 22-26, 2003), Lecture Notes in Computer Science, vol. 2702, Springer-Verlag, Berlin Heidelberg (2003), pp. 81-85.

[Golb04] Golbreich, C. Combining Rule and Ontology Reasoners for the Semantic Web. In: Antoniou, G., Boley, H. (eds.): Proceedings of the 3rd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004) (Hiroshima, Japan, November 8, 2004), Lecture Notes in Computer Science, vol 3323, Springer-Verlag, Berlin Heildelberg (2004), pp. 6-22.

[Grub93] Gruber, T.R. A translation approach to portable ontologies. In: Knowledge Acquisition, vol. 5, no. 2, June 1993. Academic Press. (1993), pp. 199-220.

[Haas06] Haase, P., Schnizler, B., Broekstra, J., Ehrig, M., van Harmelen, F., Menken, M., Mika, P., Plechawski, M., Pyszlak, P., Siebes, R., Staab, S., Tempich, C. Bibster – A Semantics Based Bibliographic Peer-to-Peer System. In: Staab, S., Stuckenschmidt (eds.): Semantic Web and Peer-to-Peer. Springer-Verlag Berlin Heidelberg (2006), pp. 349-363.

[Hafe05] Hafenrichter, B., Kießling, W. Optimization of Relational Preference Queries. In: Williams, H. E., Dobbie, G. (eds.): Proceedings of the 16th Australasian Database Conference (ADC 2005) (Newcastle, Australia, January 31st - February 3rd 2005), Conferences in Research and Practice in Information Technology, vol. 39 (2005), pp. 175-184.

[Harv05] Harvey, T., Decker K., Carberry, S. Multi-Agent Decision Support Via User Modeling. In: Aarts, H., Westra, J. (eds.): Proceedings of the 4th International Conference on Autonomous Agent and Multi-Agent Systems (AAMAS 2005) (Utrecht, Netherlands, July 25-29, 2005), ACM Press, New York, NY (2005), pp. 222-229.

[Henz02] Henze, N., Nejdl, W. Knowledge Modeling for Open Adaptive Hypermedia. In: De Bra, P., Brusilovsky P., Conejo, R. (eds.): Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002)

(Malaga, Spain, May 29-31, 2002), Lecture Notes in Computer Science, vol. 2347, Springer Verlag, Berlin Heidelberg (2002), pp. 174-183.

[Henz03] Henze, N. From Web-Based Educational Systems to Education on the Web: On the Road to the Adaptive Web. In: Palade, V., Howlett, R.; Jain, L. (eds.): Proceedings of the 7th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2003) (Oxford, UK, September 3-5, 2003), Lecture Notes in Artificial Intelligence, vol. 2774, Springer-Verlag, Berlin Heidelberg (2003), pp. 297-303.

[Herb03] Herbstritt, M., Becker, B. Conflict-Based Selection of Branching Rules. In: Giunchiglia, E., Tacchella, A. (eds.): Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003) (Santa Margherita Ligure, Italy, May 5-8, 2003), Selected Revised Papers, Lecture Notes in Computer Science, vol. 2919, Springer-Verlag, Berlin Heidelberg (2003), pp. 441-451.

[Hris03] Hristova, N., O'Hare, G.M.P., Lowen, T.D. Agent-Based Ubiquitous Systems: 9 Lessons Learnt. In: Proceedings of the Workshop on System Support for Ubiquitous Computing (UbiSys) collocated with the 5th International Conference on Ubiquitous Computing, (UbiComp 2003) (Seattle, Washington, USA, October 12, 2003) [En ligne]. Disponible sur : http://ubisys.cs.uiuc.edu/papers/the9lessons.pdf. (Consulté en Septembre 2006).

[Hris04] Hristova, N., O'Hare, G. Ad-me: wireless advertising adapted to the user location, device and emotions. In: Proceedings of 37th Annual Hawaii International Conference on System Sciences (HICSS37), Minitrack on Mobile Distributed Information Systems (MDIS) (Hawaii, Janvier 5–8, 2004), part of the Software Technology Track, IEEE Computer Society Press (2004), pp. 1-10.

[Idre04] Idreos, S., Tryfonopoulos, C., Koubarakis M., Drougas, Y. Query Processing in Super-Pair Networks with Languages Based on Information Retrieval: The P2P-DIET Approach. In: Lindner, W., Masiti, M., Türker, C., Tzitzikas, Y., Vakali, A. (eds.): Proceedings of the 9th International Conference on Extending Database Technology. (EDBT 2004) (Heraklion – Crete Greece, March 14 - 18, 2004), Lecture Notes in Computer Science, vol. 3268, Springer-Verlag, Berlin Heidelberg (2004), pp. 496-505.

[Indu03a] Indulska, J., Robinson, R., Rakotonirainy, A., Henricksen, K. Experiences in Using CC/PP in Context-Aware Systems. In: Chen, M.-S., Chrysance, P.K., Sloman, M. Zaslavsky, A. (eds.): Proceedings of the 4th International Conference on Mobile Data Management (MDM 2003) (Melbourne, Australia, January 21-24, 2003), Lecture Notes in Computer Science, vol. 2574. Springer-Verlag, Berlin Heidelberg (2003), pp. 247-261.

[Indu03b] Indulska, J., McFadden, T., Kind, M., Henricksen, K. Scalable Location Management for Context-Aware Systems. In: Stefani, J.B., Demeure, I.M., Hagimont, D. (eds.): Proceedings of the 4th IFIP WG6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2003) (Paris, France, November 17-21, 2003), Lecture Notes in Computer Science, vol. 2893, Springer-Verlag, Berlin Heidelberg (2003), pp. 224-235.

[Jean06] Jean, S., Pierra, G., Ait-Ameur, Y. Domain Ontologies : A Database-Oriented Analysis. In: Cordeiro, J., Pedrosa, V., Encarnaçao, B., Filipe, J. (eds.): Proceedings

of the 2nd International Conference on Web Information Systems and Technologies (Webist 2006) (Setubal, Portugal, April 11-13, 2006), Insticc Press , Portugal (2006), pp. 341-351.

[Jung05] Jung, E. C., Sato, K A Framework of Context-Sensitive Visualization for User-Centered Interactive Systems. In: Ardissono, L., Brna, P., Mitrovic, A. (eds.): Proceedings of the 10th International Conference on User Modeling (UM 2005) (Edinburgh, Scotland, UK, July 24-29, 2005), Lecture Notes in Computer Science, vol. 3538, Springer-Verlag, Berlin Heidelberg (2005), pp. 423-427.

[Kama04] Kamara, L., Pitt, B., Sergot, M. Norm Aware Agents for Ad Hoc Networks: A position paper. In: Proceedings of the Workshop on Agents for Ubiquitous Computing (UbiAgents04) (Columbia University, New York City, USA July 20, 2004) in conjunction with AAMAS2004 [En ligne]. Disponible sur: http://www.ift.ulaval.ca/~mellouli/ubiagents04/ (Consulté en Septembre 2006).

[Kass05] Kassab, R., Lamirel, J.C., Nauer, E. Une nouvelle approche pour la modélisation du profil de l'utilisateur dans les systèmes de filtrage d'information basés sur le contenu : le modèle de filtre détecteur de nouveauté. Actes de CORIA 2005 (Grenoble, France, 9-11 mars, 2005) pp. 185-200.

[Kech06] Kechid, S., Drias, H. Accès personnalisé à des multiples serveurs d'informations. Actes de CORIA 2006 (Lyon, France, 15-17 mars, 2006), pp. 249-254.

[Kieß05] Kießling, W. Preference Queries with SV-Semantics. In: Haritsa, J.R., Vijayaraman, T. M. (eds.): Proceedings of the 11th International Conference on Management of Data Advances in Data Management (COMAD 2005) (Goa, India, January 6-8, 2005) [En ligne] Computer Society of India (2005), pp. 15-26. Disponible sur :

http://comad2005.persistent.co.in/COMAD2005Proc/pages015-026.pdf (Consulté en Octobre 2006).

[Kirs06] Kirsch-Pinheiro, M. Adaptation Contextuelle et Personnalisée de l'information de Conscience de Groupe au sein des Systèmes d'Information Coopératifs. Thèse de doctorat, Université Joseph Fourier, Grenoble, 29 Septembre 2006.

[Koch02] Koch, N., Wirsing, M. The Munich Reference Model for Adaptive Hypermedia Applications. In: De Bra, P., Brusilovsky, P., Conejo, R. (eds.): Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002) (Malaga, Spain, May 29-31, 2002), Lecture Notes in Computer Science, vol. 2347, Springer-Verlag, Berlin Heidelberg (2002), pp. 213–222.

[Koch04] Koch, F., Rahwan, I. Classification of Agent-based Mobile Assistant. In: Proceedings of the Workshop on Agents for Ubiquitous Computing (UbiAgents04) (Columbia University, New York City, USA July 20, 2004) in conjunction with AAMAS2004 [En ligne]. Disponible sur: http://www.ift.ulaval.ca/~mellouli/ubiagents04/ (Consulté en Septembre 2006).

[Kokk04] Kokkinidis, G., Christophides, V. Semantic Query Routing and Processing in P2P Database System: The ICS-FORTH SQPair Middleware. In: Lindner, W., Masiti, M., Türker, C., Tzitzikas, Y., Vakali, A. (eds.): Proceedings of the 9th International Conference on Extending Database Technology. (EDBT 2004) (Heraklion – Crete Greece, March 14 - 18, 2004), Lecture Notes in Computer Science, vol. 3268, Springer-Verlag, Berlin Heidelberg (2004), pp. 486-495.

[Kokk06] Kokkinidis, G., Lefteris, S., Christophides, V. Query Processing in RDF/S-Based P2P Database Systems. In: Staab, S., Stuckenschmidt (eds.): Semantic Web and Peer-to-Peer. Springer-Verlag Berlin Heidelberg (2006), pp. 59-87.

[Kolo04] Koloniari, G., Pitoura, E. Content-Based Routing of Path Queries in Pair-to-Pair Systems. In: Lindner, W., Masiti, M., Türker, C., Tzitzikas, Y., Vakali, A. (eds.): Proceedings of the 9th International Conference on Extending Database Technology. (EDBT 2004) (Heraklion – Crete Greece, March 14 - 18, 2004), Lecture Notes in Computer Science, vol. 3268, Springer-Verlag, Berlin Heidelberg (2004), pp. 29-47.

[Korh03] Korhonen, J., Isto, P. Practical Experiences in Developing Ontology-Based Multi-Agent System. In: Abramowicz, W., Klein, G. (eds): Proceedings of the 6th International Conference on Business Information Systems (BIS 2003) (Colorado Springs, USA, June 4-6, 2003) (2003), pp. 147-152.

[Kost03] Kostadinov, D. Personnalisation de l'information et gestion des profils utilisateurs. [En ligne] Mémoire de DEA PRiSM, Versailles, 2003. Disponible sur : http://belzebuth.prism.uvsq.fr/apmd_public/Publications/Rapports/Personnalisation%20de%20l%20information%20et%20gestion%20des%20profils%20utilisateurs_Dimitre%20Kostadinov.pdf (Consulté en Octobre 2006)

[Koth97] Kothari, N. AGENTOS – un Java Based Mobile Agent Système. ICS Honors Project Final Report. Information et Computer Science, University de California, Irvine, 1997 [En ligne]. Disponible sur : http://netrecherche.ics.uci.edu/Previous_recherche_projects/agentos/doc/nikhil-final-report.pdf. (Consulté en January 2004).

[Kotz99] Kotz, D., Gray, R.S. Mobile Agents and the Future of the Internet. ACM Operating System Review, 1999, vol. 33, no. 3, July 1999, pp. 7-13.

[Kout04] Koutrika, G., Ioannidis, Y.E. Personalization of Queries based on User Preferences. In: Bosi, G., Brafman, R.I., Chomicki, J., Kießling, W. (eds.): Proceedings of Preferences: Specification, Inference, Applications (Preferences 2004) (June 27th – July 2nd 2004) Dagstuhl Seminar Proceedings 04271 IBFI, Schloss Dagstuhl, Germany 2006 [En ligne]. Disponible sur: http://drops.dagstuhl.de/opus/volltexte/2006/403/pdf/04271.KoutrikaGeorgia.Paper.403.pdf (Consulté en Octobre 2006).

[Kuru04] Kurumatani, K. Mass User Support for Social Coordination among Citizen in a Real Environnement. In: Chen, S-H., Ohuchi, A. (eds.): Proceedings of the International Workshop on Multi-Agent for Mass User Support. (MAMUS 2003) (Acapulco, Mexico, August 10, 2003), Lecture Notes in Artificial Intelligent, vol. 3012. Springer-Verlag, Berlin Heidelberg (2004), pp. 1–16.

[Laer04] Laera, L., Tamma, V., Bench-Capon, T., Semeraro, G. SweetProlog: A system to Integrate Ontologies and Rules. In: Antoniou, G., Boley, H. (eds.): Proceedings of the 3rd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004) (Hiroshima, Japan, November 8, 2004), Lecture Notes in Computer Science, vol 3323, Springer-Verlag, Berlin Heildelberg (2004), pp. 188-193.

[Lech05] Lech, T., Wienhofen, L. AmbieAgents: A Scalable Infrastructure for Mobile and Context-Aware Information Services. In: Aarts, H., Westra, J. (eds.): Proceedings of

the 4th International Conference on Autonomous Agent and Multi-Agent Systems (AAMAS 2005) (Utrecht, Netherlands, July 25-29, 2005), ACM Press, New York, NY (2005), pp. 625-631.

[Leml04] Lemlouma, T. Architecture de Négociation et d'Adaptation de Services Multimédia dans des Environnements Hétérogènes. Thèse de Doctorat. Institut National Polytechnique de Grenoble, Grenoble, June 2004.

[LiLi04] Li, J., Li, J., Shi, S. A Query-Aware Routing Algorithm in Sensor Network. In: Jin, H., Gao, G.R., Xu, Z., Chen, H. (eds.): Proceedings of the IFIP International Conference on Network and Parallel Computing (NPC 2004) (Wuhan, China, October 18-20, 2004), Lecture Notes in Computer Science, vol. 3222, Springer-Verlag, Berlin Heidelberg (2004), pp. 528-535.

[LinL04] Lin, F.C., Liu, H.H. MASPF: Searching the Shortest Communication Path with the Guarantee of the Message Delivery between Manager and Mobile Agent. In: Yang, L., Guo, M., Gao, G.R., Jha, N.K. (eds.): Proceedings of the Conference on Embedded and Ubiquitous Computing (EUC 2004) (Aizu-Wakamatsu City, Japan, August 25-27, 2004), Lecture Notes in Computer Science, vol. 3207, Springer-Verlag, Berlin Heidelberg (2004), pp. 755-764.

[LiZo04a]Li, Y., Zou, F., Ma, F., Li, M. pXRepository: A Peer to Peer XML Repository for Web Service Discovery. In: Jin, H., Pan, Y., Xiao, N., Sun, J. (eds.): Proceedings of the 3rd International Conference and Cooperative (GCC 2004) (Wuhan, 430074, China. October 20-24, 2004), Lecture Notes in Computer Science, vol. 3251, Springer-Verlag, Berlin Heidelberg (2004), pp 137-144.

[LiZo04b]Li, Y., Zou, F., Ma, F., Li., M. eXChord: Build a Distributed Repository for Web Service Discovery Based on Peer-to-Peer Network. In: Jin, H., Gao, G.R., Xu, Z., Chen, H. (eds.): Proceedings of the Conference on Network and Parallel Computing (NPC 2004) (Wuhan, China, October 18-20, 2004), Lecture Notes in Computer Science, vol. 3222, Springer-Verlag, Berlin Heidelberg (2004), Chapter: pp. 175.

[Llad06] Lladó, E., Salamanca, I. Xarop, a Semantic Peer-to-Peer System for a Virtual Organisation. In: Staab, S., Stuckenschmidt (eds.): Semantic Web and Peer-to-Peer. Springer-Verlag Berlin Heidelberg (2006), pp. 335-347.

[Lope02] Lopez y Lopez, F., Lucj, M., d'Inverno, M. Constraining Autonomy through Norms. In: Proceedings of the 1st International Joint Conferences on Autonomous Agent and Multi-Agent System (AAMAS 2002) (Bologna, Italy, July 15-19, 2002), ACM Press, New York, NY (2002), pp. 674-681.

[Mage99] Magedanz T., Baumer C., Breugst M., Choy S. Grasshopper – A universal Agent Platform Based on OMG MASIF and FIPA Standards. Agents Technologies in Europe - ACTS Activities. September, 1999 [En ligne]. Disponible sur : http://www.cordis.lu/infowin/acts/analysys/products/thematic/agents/toc.htm. (Consulté en Septembre 2006).

[Math04] Matheus, C.J. SWRLp: An XML-Based SWRL Presentation Syntaxis. In: Antoniou, G., Boley, H. (eds.): Proceedings of the 3rd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004) (Hiroshima, Japan, November 8, 2004), Lecture Notes in Computer Science, vol 3323, Springer-Verlag, Berlin Heildelberg (2004), pp. 194-199.

[Maxi05] Maximilien, E., Singh, M. Agent-Based Trust Model Involving Multiple Qualities. In: Aarts, H., Westra, J. (eds.): Proceedings of the 4th International Conference on Autonomous Agent and Multi-Agent Systems (AAMAS 2005) (Utrecht, Netherlands, July 25-29, 2005) ACM Press, New York, NY (2005), pp. 519-526.

[Mcke04] Mckenzie, C., Gray, P., Preece, A. Extending SWRL to Express Fully Quantified Constraints. In: Antoniou, G., Boley, H. (eds.): Proceedings of the 3rd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004) (Hiroshima, Japan, November 8, 2004), Lecture Notes in Computer Science, vol 3323, Springer-Verlag, Berlin Heildelberg (2004), pp. 139-154.

[Medi03] Medina-Medina, N., Molina-Ortiz, F., García-Cabrera, L., Parets-Llorca, J. Personalized Guided Routes in an Adaptive Evolutionary Hypermedia System. In: Moreno-Díaz, R., Pichler, F. (eds.): Proceedings of the 9th International Workshop on Computer Aided Systems Theory on Computer Aided Systems Theory (EUROCAST 2003) (Las Palmas de Gran Canaria, Spain, February 24-28, 2003), Lecture Notes in Computer Science, vol. 2809, Springer Verlag, Berlin Heidelberg (2003), pp. 196-207.

[MeiL04] Mei, J., Liu, S., Yue, A., Lin, Z. An extension to OWL with General Rules. In: Antoniou, G., Boley, H. (eds.): Proceedings of the 3rd International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004) (Hiroshima, Japan, November 8, 2004), Lecture Notes in Computer Science, vol 3323, Springer-Verlag, Berlin Heildelberg (2004), pp. 155-169.

[Murr00] Murray, T., Piemonte, J., Khan, S., Shen, T., Condit, C. Evaluating the Need for Intelligence in an Adaptive Hypermedia System. In: Gauthier, G., Frasson, C., Frasson, C. (eds): Proceedings of the 5th International Conference on Intelligent Tutoring Systems (ITS 2000) (Montréal, Canada, June 19-23, 2000), Lecture Notes in Computer Science, vol 1839, Springer Verlag, Berlin Heidelberg (2000), pp. 373-382.

[Nage99] Nagendra Prasad, M.V, MACCrthy, J. A Multi-Agent System for Meting Out Influence in an Intelligent Environnement. In: Proceedings of the 11th Conference on Innovation Applications of Artificial Intelligence (IAAI' 99) (Orlando, Florida, USA, July 18-22, 1999), AAAI Press / The MIT Press (eds.) (1999) [En ligne]. Disponible sur: http://seattleweb.intel-recherche.net/people/mACCrthy/MusicFX-IAAI99.PDF. (Consulté en Janvier 2004), pp. 884-890.

[Niet04] Nieto-Carvajal, I., Botia, J.A., Ruiz, P.M., Gomez-Skarmeta, A.F. Implementation and Evaluation of a Location-Aware Wireless Multi-Agent System. In: Yang, L., Guo, M., Gao, G.R., Jha, N.K. (eds.): Procedings of the Embedded and Ubiquitous Computing (EUC 2004) (Aizu-Wakamatsu City, Japan, August 25-27, 2004), Lecture Notes in Computer Science, vol. 3207, Springer-Verlag, Berlin Heidelberg (2004), pp. 528-537.

[Odel00] Odell, J., Van Dyke Parunak, H., Bauer, B. Representing Agent Interaction Protocols in UML. In: P. Ciancarini and M.J. Wooldridge (eds.): Proceedings of the 1st International Workshop on Agent Oriented Software Engineering (AOSE 2000) (Limerick, Ireland, June 10, 2000), Lecture Notes in Computer Science, vol. 1957, Springer-Verlag Berlin Heidelberg (2001), pp. 121-140.

[OHar02] O'Hare, G., O'Grady, M. Addressing Mobile HCI Need through Agents. In: Paterno, F. (ed.): Proceedings of the 4th International Symposium on Human Computer

Interaction with Mobile Devices and Services (MobileHCI'02) (Pisa, Italy, September 18-20, 2002), Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, vol. 2411 (2002), pp. 311-314.

[Pant02] Panti, M., Penserini, L., Spalazzi, L. A Multi-Agent System based on the P2P model to Information Integration. In: Proceedings of the 1st International joint Conference on Autonomous Agents and Multi-Agent Systèmes (AAMAS 2002). (Bologna, Italy. 2002) [En ligne]. Disponible sur: http://www.agentcities.org/EUNET/Projects/acnet_proj_38.pdf (Consulté en Septembre 2006)

[PanC03] Pan, J., Cranefield, S., Carter, D. A Lightweight Ontology Repository. Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003) (Melbourne, Victoria, Australia, July 14-18, 2003), ACM Press, New York, NY (2003), pp. 632-638.

[Park04] Park, J., Barber, S. Finding Information Sources for Model Sharing in Open Multi-Agent System. In: Proceedings of the Workshop on Agents for Ubiquitous Computing (UbiAgents04) (July 20, 2004, Columbia University, New York City) [En ligne]. Disponible sur: http://www.ift.ulaval.ca/~mellouli/ubiagents04/ (Consulté en Septembre 2006)

[Pirk04] Pirker, M., Berger M., Watzke, M. An approach for FIPA Agent Service Discovery in Mobile Ad Hoc Environnements. In: Proceedings of the Workshop on Agents for Ubiquitous Computing (UbiAgents04) (July 20, 2004, Columbia University, New York City) [En ligne]. Disponible sur: http://www.ift.ulaval.ca/~mellouli/ubiagents04/ (Consulté en Septembre 2006).

[Pitt05] Pittarello, F. Context-Based Management of Multimedia Documents in 3D Navigational Environments. In: Candan, K.S., Celentano, A. (eds.): Proceedings of the 11th International Workshop on Advances in Multimedia Information Systems (MIS 2005) (Sorrento, Italy, September 19-21, 2005), Lecture Notes in Computer Science, vol. 3665, Springer-Verlag, Berlin Heidelberg (2005), pp. 146-162.

[Pogg03] Poggi, A., Rimassa, G., Turci, P., Odell, J., Mouratidis, H., Manson, G. Modelling Deployment and Mobility Issues in Multi-agent Systems Using AUML. In: Giorgini, P., Mûller, J.P., Odell, J. (eds.): Proceedings of the 4th International Workshop on Agent Oriented Software Engineering (AOSE 2003) (Melbourne, Australia - July 15, 2003), Lecture Notes in Computer Science, vol. 2935. Springer-Verlag, Berlin Heidelberg (2004), pp. 69-84.

[Quin98] Quintero, A., Rueda, S., Ucrós, M.E. : Agentes y Sistemas Multiagente : Integración de Conceptos Básicos. [En ligne] Disponible sur : http://agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm (Consulté en Novembre 2006).

[Raad02] Raad, H., Causse, B. Modelling of an Adaptive Hypermedia System Based on Active Rules. In: Cerri, S., Gouardères, G., Paraguaçu, F. (eds.): Proceedings of the 6th International Conference Intelligent Tutoring Systems (ITS 2002) (Biarritz, France and San Sebastian, Spain, June 2-7, 2002), Lecture Notes in Computer Science, vol. 2363, Springer-Verlag, Berlin Heidelberg (2002), pp. 149-157.

[Rahw04] Rahwan, T., Rahwan, T., Rahwan, I., Ashri, R. Agent-Based Support for Mobile Users Using AgentSpeak (L). In: Giorgini P., Henderson-Sellers B., Winikoff, M. (eds.): Proceedings of the Workshop on Agent-Oriented Information Systems (AOIS 2003) (Melbourne, Australia, July 14, 2003 - Chicago, USA, October 13, 2003), Lecture Notes in Artificial Intelligence, vol. 3030 Springer-Verlag, Berlin Heidelberg (2004), pp. 45-60.

[Ramp03] Ramparany, F., Boissier, O., Brouchoud H. Cooperating Autonomous Smart Devices. In: Proceedings of the Smart Objects Conference (sOc'2003). Grenoble, France, May 15-17, 2003. (2003), pp.182-185.

[Raza02] Raza Abidi, S. Designing Adaptive Hypermedia for Internet Portals: A Personalization Strategy Featuring Case Base Reasoning with Compositional Adaptation. In: Garijo, F., Riquelme Santos, J., Toro, M. (eds.): Proceedings of the 8th Ibero-American Conference on AI Advances in Artificial Intelligence (IBERAMIA 2002) (Seville, Spain, November 12-15, 2002), Lecture Notes in Computer Science, vol. 2527, Springer-Verlag, Berlin Heidelberg (2002), pp. 60-69.

[Rohm00] Röhm, U., Böhm, K., Schek, H. OLAP Query Routing and Physical Design in a Database Cluster. In: Zaniolo, C., Lockemann, P.C., Scholl, M.H., Grust T. (eds.): Proceedings of the 7th Conference on Extending Database Technology (EDBT 2000) (Konstanz, Germany, March 27-31, 2000), Lecture Notes in Computer Science, vol. 1777, Springer-Verlag, Berlin Heidelberg (2000), pp. 254-268.

[Ross03] Ross, R.G. The Business Rules Manifesto. Version 2.0, November 1, 2003 [En ligne]. Disponible sur: Business Rules Group: http://www.BusinessRulesGroup.org.

[Sash04] Sashima, A., Izumi, N., Kurumatani, K. CONSORTS: A Multi-agent Architecture for Service Coordination in Ubiquitous Computing. In: Chen, S-H., Ohuchi, A. (eds.): Proceedings of the International Workshop on Multi-Agent for Mass User Support. (MAMUS 2003) (Acapulco, Mexico, August 10, 2003), Lecture Notes in Artificial Intelligence, vol. 3012. Springer-Verlag, Berlin Heidelberg (2004), pp. 190–216.

[Schw05] Schwinger, W., Grün, Ch., Pröll, B., Retschitzegger, W., Schauerhuber, A. Context-awareness in Mobile Tourisme Guides – A Comprehensive Survey. Rapport Technique. Johannes Kepler University Linz (2005) 20 pages. (Consulté en Octobre 2006) Disponible sur : http://www.wit.at/people/schauerhuber/publications/contextAwareMobileTourismGuides_TechRep0507.pdf

[Sear79] Searle, J. Expression and Meaning: Studies in the Theory of Speech Acts. Cambridge University Press. New York, 1979.

[Seef04] Seefelder de Assis, P., Schwabe, D. A General Meta-model for Adaptive Hypermedia Systems. In: De Bra, P., Nejdl, W. (eds.): Proceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004) (Eindhoven, The Netherlands, August 23-26, 2004), Lecture Notes in Computer Science, vol. 3137, Springer Verlag, Berlin Heidelberg (2004), pp. 433-436.

[Ship81] Shipman, D. The Functional Data Model and the Data Languages DAPLEX. In: ACM Transaction on Database Systems, vol 6, no. 3 (1981), pp. 140-173.

[Shiz04] Shizuka, M., MA, J., Lee, J., Miyoshi, Y., Takata, K. A P2P Ubiquitous System for Testing Network Programs. In: Yang, L., Guo, M., Gao, G.R., Jha, N.K. (eds.):

Proceedings of the Embedded and Ubiquitous Computing (EUC 2004) (Aizu-Wakamatsu City, Japan, August 25-27, 2004), Lecture Notes in Computer Science, vol. 3207, Springer-Verlag, Berlin Heidelberg (2004), pp. 1004-1013.

[Sibe04] Siberski W., Thaden, U. A simulation Framework for Schema-Based Query Routing in P2P Network. In: Lindner, W., Masiti, M., Türker, C., Tzitzikas, Y., Vakali, A. (eds.): Proceedings of the 9th International Conference on Extending Database Technology. (EDBT 2004). (Heraklion – Crete Greece, March 14 - 18, 2004), Lecture Notes in Computer Science, vol. 3268, Springer-Verlag, Berlin Heidelberg (2004), pp. 436-445.

[Smit80] Smith, R.G. The contract net protocol: high level communication and control in a distributed problem solver. In: IEEE Transactions on Computers, vol. 29, no. 12 (1980), pp. 1104-1113.

[Stuc06] Stuckenschmidt, H., van Harmelen, F., Siberski, W., Staab, S. Peer-to-Peer and Semantic Web. In: Staab, S., Stuckenschmidt (eds.): Semantic Web and Peer-to-Peer. Springer-Verlag Berlin Heidelberg (2006), pp. 1-17.

[Sugu99] Suguri, H., Kodama, E., Miyazaki, M. Implementation of FIPA Ontology Service [En ligne]. Disponible sur : http://ias.comtec.co.jp/ap/comtec-ontology-service.pdf (Consulté en Février 2004).

[Taka03] Takahashi, K., Amamiya, S., Iwao, T. An Agent-based Framework for Ubiquitous Systems. In: Workshop on Challenges in Open Agent Systèmes '03 (Challenge03) [En ligne]. Melbourne, Australia, July 14-15 2003. (2003). Disponible sur: http://www.agentcities.org/Challenge03/articles.php (Consulté en Janvier 2004).

[Talm05] Talman, S., Hadad, M., Gal, Y., Kraus, S. Adapting to Agents' Personalities in Negotiation. In: Aarts, H., Westra, J. (eds.): Proceedings of the 4th International Conference on Autonomous Agent and Multi-Agent Systems (AAMAS 2005) (Utrecht, Netherlands, July 25-29, 2005), ACM Press, New York, NY (2005), pp. 383-389.

[Tami06] Tamine, L., Bahsoun, W. Définition d'un profil multidimensionnel de l'utilisateur. Actes de CORIA 2006 (Lyon, France, 15-17 mars, 2006), pp. 225-236.

[Thil04] Thilliez M., Delot T. Evaluating Location Dependent Queries Using ISLANDS. In: Ramos, F., Unger, H., Larios, V. (eds.): Proceedings of the Symposium on Advanced Distributed Systems (ISSADS 2004) (Guadalajara, Mexico, January 25-30, 2004), Lecture Notes in Computer Science, vol. 3061, Springer-Verlag, Berlin Heidelberg (2004), pp. 126-136.

[Titk04] Titkov, L., Poslad, S. Supporting privacy for U-commerce tourism services. In: Proceedings of the Workshop on Agents for Ubiquitous Computing (UbiAgents04) (Columbia University, New York City, USA July 20, 2004) in conjunction with AAMAS 2004 [En ligne]. Disponible sur: http://www.ift.ulaval.ca/~mellouli/ubiagents04/ (Consulté en Septembre 2006)

[Vdov06] Vdovjak, R., Houben, G-J., Stuckenschmidt, H., Aerts, A. RDF and Traditional Query Architectures. In: Staab, S., Stuckenschmidt (eds.): Semantic Web and Peer-to-Peer. Springer-Verlag Berlin Heidelberg (2006), pp. 41-58.

[Vill02] Villanova, M. Adaptabilité dans les systèmes d'information sur le web : Modélisation et mise en œuvre de l'accès progressif. Thèse Doctorale (In French), INPG, France, 2002.

[Weib03] Weib, G., Rovatsos, M., Nickles, M. Capturing Agent Autonomy in Roles and XML. In: Proceedings of the Conference on Autonomous Agents and Multi-Agent System (AAMAS 2003) (Melbourne, Australia, July 14-18, 2003), ACM Press, New York, NY, USA (2003), pp. 105-112.

[Weny01] Wenyin, L., Chen, Z., Li, M.,Zhang, H. A Media Agent for Automatically Building a Personalized Semantic Index of the Web Media Objects. Journal of the American Society for Information Science, vol. 52, no.10 (2001), pp. 853-855.

[Wool95] Wooldridge, M., Jennings, N.R. Intelligent Agents: Theory and Practice. In: The Knowledge Engineering Review, vol. 10, no. 2 (1995), pp. 115-152.

[XuLi99] Xu, J., Lim, E., Ng, W.K. Cluster-Based Database Selection Techniques for Routing Bibliographic Queries. In: Bench-Capon, T., Soda, G., Min Tjoa, A. (eds.): Proceedings of the 10th International Conference and Workshop on Database and Expert Systems Applications (DEXA 99) (Florence, Italy, August 30 - September 3, 1999), Lecture Notes in Computer Science, vol. 1677, Springer-Verlag, Berlin Heidelberg (1999), pp.100-109.

[XuZh03] Xu, L., Zhou, S., Zhao, K., Qian, W., Zhou, A. PairBus: A Middleware Framework towards Interoperability among P2P Data Sharing Systems. In: Li, M., Sun, X-H., Deng, Q., Ni, J. (eds.): Proceeding of the 2nd International Workshop in Grid and Cooperative Computing (GCC 2003) (Shanghai, China, December 7- 10, 2003), Lecture Notes in Computer Science, vol. 3032. Springer-Verlag, Berlin Heidelberg (2003), pp. 277-284.

[Yang03] Yang, K., Galis, A., Todd, C. Policy-driven Mobile Agents for Context-aware Service in Next Generation Networks. In: Proceedings of the 5th International Conference on Mobile Agents for Telecommunications (MATA'03) (Marrakech, Morocco, October 8-10, 2003). [En ligne]. Disponible sur : http://context.upc.es/publications.htm (Consulté en Septembre 2006).

[Yang04] Yang, D., Xu, L., Cai, W., Zhou, S., Zhou, A. Efficient Query Routing for XML Documents Retrieval in Unstructured Peer to Peer Networks. In: Yu, J.X., Lin, X., Lu, H., Zhang, Y. (eds.): Proceedings of the 6th Asia Pacific Web Conference (APWeb 2004) (Hangzhou, China, April 14-17, 2004), Lecture Notes in Computer Science, vol. 3007, Springer-Verlag, Berlin Heidelberg (2004), pp. 217-223.

[Yude05] Yudelson, M., Gavrilova, T., Brusilovsky, P. Towards User Modeling Meta-ontology. In: Ardissono, L., Brna, P., Mitrovic, A. (eds.): Proceedings of the 10th International Conference on User Modeling (UM 2005) (Edinburgh, Scotland, UK, July 24-29, 2005), Lecture Notes in Computer Science, vol. 3538, Springer-Verlag, Berlin Heidelberg (2005), pp. 448-452.

[Zaca05] Zacarias, M., Caetano, A., Pinto, S., Tribolet, J. Modeling Contexts for Business Process Oriented Knowledge Support. In: Althoff, K.D., Dengel, A., Bergmann, R., Nick, M., Roth-Berghofer, T. (eds.): Proceedings of the 3rd Conference on Professional Knowledge Management - Experiences and Visions (WM 2005) (Kaiserslautern, Germany, April 10-13, 2005), DFKI (2005), pp. 389-396.

[Zemi05] Zemirli, N., Lechani Tamine, L., Boughanem, M. Accès personnalisé à l'information : Proposition d'un profil utilisateur multidimensionnel. In: Proceedings of the 7th International Symposium on Programming and Systems (ISPS'2005) (Algiers, Algeria May 9-11, 2005) [En ligne]. Disponible sur: http://www.isps2005.dz/proceedings/papers/3-244.pdf (Consulté en Octobre 2006).

[Zhug04] Zhuge, H., Liu, J., Feng, L., He, C. Semantic-Based Query Routing and Heterogeneous Data Integration in Peer to Peer Semantic Link Networks. In: Bouzeghoub, M., Goble, C.A., Kashyap, V., Spaccapietra, S. (eds.): Proceedings of the 1st International IFIP Conference on Semantics of a Networked World and Semantics for Grid Databases (ICSNW 2004) (Paris, France, June 17-19, 2004), Lecture Notes in Computer Science, vol. 3226, Springer-Verlag, Berlin Heidelberg (2004), pp. 91-107.

[JADE06] JADE-LEAP: http://jade.tilab.com/

[W3Ca04] CC/PP: http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-200401

[W3Cb05] OWL: http://www.w3.org/TR/webont-req/ (Dernier Accès : Mars 2005)

[FIPA04] FIPA ACL: http://www.fipa.org/specs/fipa00061/SC00061G.html (Dernier Accès : Août 2005)

RESUME :

L'informatique ubiquitaire est un paradigme récent dont l'objectif est de permettre aux utilisateurs de consulter des données n'importe quand, n'importe où, notamment à l'aide de *Dispositifs Mobiles* (*DM*). Dans ce cadre, garantir l'accès par des utilisateurs nomades aux *Systèmes d'Information* (*SI*) à travers divers dispositifs, ainsi que l'adaptation de l'information aux préférences de l'utilisateur nomade et au contexte d'utilisation de la session en cours, sont deux problèmes liés non encore résolus. En effet, les utilisateurs nomades de *SI* peuvent être exposés à une grande quantité d'informations qui n'est pas toujours pertinente ni même simplement supportée par leurs *DM*. A des fins d'adaptation, un *SI* doit donc disposer de données sur le contexte d'utilisation – notion qui englobe les conditions temporelles, spatiales, matérielles, etc. dans lesquelles l'utilisateur accède au *SI* –, mais également sur les préférences de l'utilisateur afin de satisfaire ses attentes à la fois en termes de fonctionnalités, de contenu, et d'affichage.

A travers deux propositions, les travaux de thèse exposés ici tentent d'apporter une réponse à cette double problématique. Tout d'abord, nous avons conçu et réalisé un framework appelé *PUMAS* qui offre à des utilisateurs nomades un accès à l'information, qui prend en compte le contexte d'utilisation. L'approche que nous avons choisie est celle des agents. Ainsi, l'architecture de *PUMAS* est composée de quatre *Systèmes Multi-Agents* (*SMA*) respectivement dédiés à la connexion aux *SI*, la communication entre les utilisateurs et les *SI*, la gestion de l'information et l'adaptation de celle-ci. Ensuite, nous avons élaboré un *Système de Gestion de Profil Contextuel* (*SGPC*) qui contribue à l'adaptation de l'information délivrée à un utilisateur nomade sur trois aspects : *i)* une formalisation de la notion de préférence de l'utilisateur qui permet de modéliser les activités accomplies dans le système, les résultats attendus de ces activités et la manière dont ces résultats sont présentés ; *ii)* un algorithme de correspondance contextuelle qui génère le profil contextuel d'un utilisateur nomade à partir du contexte d'utilisation ; *iii)* un mécanisme qui gère les conflits pouvant survenir entre les préférences de l'utilisateur. Enfin, le *SGPC* a été intégré à *PUMAS* au sein du *SMA* dédié à l'adaptation de l'information.

MOTS-CLES : Système d'Information basé sur le Web, Système Multi-Agents, Environnement nomade, Adaptation, Préférences de l'utilisateur, Profil contextuel de l'utilisateur.

ABSTRACT :

Ubiquitous Computing is a recent paradigm whose objective is to allow users to access data any time, anywhere, in particular using *Mobile Devices* (*MD*). In this context, insuring the access of nomadic users to *Information Systems* (*IS*) through different devices, as well as the adaptation of information to the preferences of the nomadic users and to the context of use of the current session, are two related problems not yet solved. Indeed, nomadic users of *IS* can be exposed to a great quantity of information which is not always relevant nor is not even simply supported by their *MD*. For adaptation purposes, an *IS* must possess data about the context of use - concept which includes the temporal, spatial and material conditions in which the user accesses *IS* -, but also about the preferences of the user in order to satisfy her/his expectations at the same time in terms of functionalities, contents, and display.

Through two proposals, this thesis tries to give an answer to this double problem. Firstly, we designed and realised a framework called *PUMAS* which provides nomadic users with an access to the information, which considers the context of use. The approach that we chose is based on agents. Thus, the architecture of *PUMAS* is composed of four *Multi-Agent Systems* (*MAS*) respectively dedicated to connection to *IS*, the communication between the users and *IS*, the information management and the adaptation of this information. Secondly, we created a *Contextual Profile Management System* (*CPMS*) which contributes to the adaptation of the information delivered to a nomadic user on three aspects: *i)* a formalization of the notion of preference of the user that allows to model the activities achieved in the system, the expected results of these activities and the way in which these results are presented; *ii)* a *Contextual Matching Algorithm* which generates the contextual profile of a nomadic user considering the context of use; *iii)* a mechanism for managing the conflicts which can occur between the preferences of the user. Finally, the *CPMS* was integrated into *PUMAS* within the *MAS* dedicated to the adaptation.

KEYWORDS: Web Information Systems, Multi-Agent System, Nomadic environment, Adaptation, User preferences, Contextual user profile.