Sommaire

Liste des figures

Introduction générale

Introduction

Les systèmes d'information actuels donnent accès à des sources de données multiples, distribuées, autonomes et potentiellement redondantes. Une des principales limites de ces systèmes est leur incapacité à discriminer les utilisateurs en fonction de leurs centres d'intérêt, de leurs préférences et de leurs contextes de requêtage, et à leur délivrer des résultats pertinents selon leurs profils respectifs. Cette limite a plusieurs conséquences pour l'utilisateur:

- Les mêmes réponses sont fournies aux mêmes requêtes quels que soient les utilisateurs qui les ont émises: les systèmes se contentant de délivrer tous les objets satisfaisant strictement les critères de la requête;
- La taille des réponses est souvent volumineuse et génère une surcharge informationnelle qui déroute ou décourage l'utilisateur dans son exploration ou sa navigation;
- La pertinence des réponses se trouve souvent réduite dans la mesure où elles ne sont pas adaptées au contexte de l'utilisateur : les objets délivrés ne sont pas forcément en adéquation avec le lieu d'émission de la requête ou le terminal utilisé pour la requête;
- La qualité de l'information délivrée est ignorée; ce qui laisse souvent l'utilisateur perplexe quant à son utilité et rend difficile la prise de décision à base de cette information.

Ces conséquences sont inhérentes aux systèmes de bases de données qui ont été conçus pour une utilisation dans des domaines d'applications fermés, où l'utilisateur connaît non seulement le schéma de la base, mais suppose aussi que tous les objets de son univers sont dans la base de données au moment de leur utilisation (hypothèse du monde fermé). De ce fait, la requête est une expression exacte de son besoin, qui désigne les objets auxquels il veut appliquer un certain traitement. Les seules variations admises sont celles liées aux mises à jour qui peuvent altérer les résultats d'une requête selon le moment de son exécution; variations admises car le monde réel de l'utilisateur est évolutif et la base de données est vue comme une succession d'états représentant ce réel.

Les systèmes d'intégration de données n'ont fait que généraliser cette approche à un ensemble de sources de données distribuées, sans remise en cause de l'hypothèse du monde fermée, prolongeant ainsi la survie des modes opératoires classiques. Par exemple, un système de médiation de données est perçu à travers son schéma global sur lequel l'utilisateur exprime ses requêtes qui sont ensuite réécrites ou décomposées pour être exécutées de façon tout à fait classique sur les sources de données participant à la médiation.

L'évolution des sources de données, leur indisponibilité temporaire ou permanente, l'ajout de nouvelles sources ne sont pas pris en compte; tout se passe comme s'il y a toujours un administrateur qui maintient la vision classique d'une base de données, sans variation de sa sémantique initiale.

Cependant, la multiplicité des sources de données, leur évolutivité et la difficulté croissante de maîtriser leurs descriptions et leurs contenus (notamment dans les architectures P2P) font émerger de nouvelles pratiques qui s'apparentent plus à celles utilisées dans les systèmes de recherche d'information (SRI). Les utilisateurs ne connaissent pas forcément les

sources de données qu'ils interrogent, leur description leurs sont inaccessibles et ils ne savent même pas si l'information qu'ils recherchent existe ou non. En conséquence, leurs requêtes ne traduisent plus un besoin précis mais une intention qui doit être affinée en fonction des sources de données disponibles dans le système d'intégration au moment de l'interrogation.

Par ailleurs, ces utilisateurs ont de nouvelles exigences telles que la prise en compte de leur localité géographique, le média utilisé pour l'expression de leurs requêtes, leurs préférences récurrentes en termes de qualité des données, de présentation des résultats, de sécurité, etc. Ainsi, si ces préférences sont prises en compte, l'exécution de la même requête, exprimée par des utilisateurs différents, ne produit pas nécessairement les mêmes résultats. C'est ce qu'on appelle un accès personnalisé à l'information.

Contribution

Pour répondre aux besoins de la personnalisation, différentes approches ont été adoptées: extension des langages de requêtes comme dans PreferenceSQL [Kieß 02], enrichissement de requêtes à l'aide de préférences définies dans un profil utilisateur [KoIo 04], sélection des sources de données en fonction de leur qualité [Naum 98]. Les deux premières approches s'inscrivent dans le cadre de l'accès à une source de données unique. La troisième approche s'inscrit dans le cadre de systèmes multisource mais ne vise que la sélection des sources selon des critères de qualité. Par ailleurs, c'est une approche statique réalisée pour un ensemble de requêtes et non pour chaque requête.

Mais aucune de ces approches ne prend en compte la personnalisation dans sa globalité, tenant compte à la fois des profils des utilisateurs (centre d'intérêt, préférences, contexte d'exécution de la requête) et des profils des sources de données (méta données décrivant leurs contenus et leurs facteurs de qualité). En effet, les bénéfices de l'accès personnalisé à l'information sont plus visibles dans un contexte distribué où la multiplicité des sources de données conduit à des résultats volumineux, souvent non pertinents et redondants. C'est le cas des systèmes de médiation qui délivrent l'ensemble des résultats possibles collectés à partir des sources qui leur sont connectées, sans évaluation de leur pertinence par rapport aux préférences de l'utilisateur. Le problème peut être encore pire dans le cas des systèmes P2P où la requête de l'utilisateur est disséminée sur le réseau pour acquérir un maximum d'informations sans tenir compte d'autres critères que ceux exprimés dans la requête elle-même.

La première étape de ce rapport consiste à étudier les différentes approches de personnalisation de l'information et dont tirer les avantages et les limites de chaque approche. La deuxième étape consiste à proposer une méthodologie de conception des systèmes d'information personnalisés dans un contexte de médiation à grande échelle où les sources de données sont évolutives et sujettes à des déconnexions temporaires ou permanentes. Dans ce contexte, l'évaluation d'une requête se fait en tenant compte des sources disponibles et de leur qualité au moment de l'évaluation de la requête tandis que la réponse fournie à l'utilisateur est sélectionnée en tenant compte de son profil.

Une étude de cas pratique a été réalisée sur l'exemple de portail étudiant personnalisé qui fournit des services de recherche personnalisée et non personnalisée d'informations, comme il inclus les fonctionnalités d'alertes automatiques pour les utilisateurs concernés par une nouveauté ou information connexe offertes par le système.

Composition du rapport

Ce rapport est structuré en cinq deux grandes parties :

La première partie est composée de deux chapitres :

- Le premier chapitre, intitulé « Systèmes d'information personnalisés et approches de la personnalisation». la première section de ce chapitre a pour objectif de présenter les avantages des systèmes d'information personnalisée relativement aux systèmes d'information classiques. La deuxième section décrit les différentes approches de personnalisation de l'information ainsi, que leurs avantages et leurs limites
- Le deuxième chapitre, intitulé « Approche de personnalisation proposée et thématiques de travail »propose une méthodologie de conception des systèmes d'information personnalisés à accès aux bases de données hétérogènes et distribuer. En outre il décrit la thématique dans laquelle s'inscrit notre mémoire.

La deuxième partie est composée de trois chapitres :

- Le premier chapitre, intitulé « PerESPERE un système de personnalisation pour le projet ESPERE » propose un système de personnalisation de l'information sur mesure pour l'application « portail étudiant » présentée dans la suite de ce rapport. Ce chapitre débute par une présentation de la plateforme utilisée pour le développement des agents logiciels composant PerEspere, pour finir par décrire l'architecture générale et la conception de notre système de personnalisation.
- Le deuxième chapitre, intitulé « MedESPERE : un système de médiation de données hétérogènes » propose une conception de notre système de médiation ainsi que quelques composants du médiateur. On présente également un cas d'utilisation afin d'illustrer les différents traitements qui doit accomplir le médiateur.
- Le troisième chapitre, intitulé « Spécification d'un Espace de Services PERsonnalisés pour l'Etudiant (ESPERE) » consiste à concevoir ESPERE (Espace de Services PERsonnalisés pour l'Etudiant). Ce chapitre est composé de deux parties. La première partie présente quelque notions qui seront utilisés pour notre application. Pour la deuxième partie, nous y dégageons les besoins techniques et nous présentons les modèles des cas d'utilisations ainsi que les modèles d'analyse des cas d'utilisation jugés prioritaires.

Première partie

Chapitre 1
Systèmes d'information personnalisés et approches de la personnalisation

Sommaire

Introduction

Un système d'information (SI) est un système composé d'un ensemble de données et de procédures dont le but est de fournir de l'information à ses utilisateurs ; les guichets automatiques délivrant des billets de transport en sont un exemple. Ces systèmes facilitent ainsi nos tâches quotidiennes. Les systèmes d'information se déclinent sous plusieurs formes ; les différences portent essentiellement sur les trois points suivants [Rosé .03].

La complexité qui peut varier du simple système informatif "question-réponse" (à une question est associée une seule réponse, par exemple une borne indiquant le prix d'un article à partir de son code-barres dans un supermarché) aux systèmes plus complexes tels que les systèmes d'aide à la supervision en milieu industriel.

L'architecture : les deux principaux types se retrouvent (i) dans les systèmes centralisés, dont la totalité des données et des procédures est localisée dans un même lieu comme le système de gestion d'une bibliothèque personnelle, (ii) et les systèmes distribués dans lesquels les données et/ou les procédures sont localisées en des lieux physiquement différents. Un système de réservation d'une agence de voyages est un exemple de système distribué dans la mesure où il doit interroger d'autres systèmes afin de connaître les disponibilités de places pour un vol ou un hôtel.

Le type d'utilisateurs visé est une donnée importante dans la conception des systèmes d'information. Cela peut varier des systèmes grands publics (par exemple une borne d'information touristique dans une ville) aux systèmes exclusivement dédiés à des experts (appliqué par exemple à la simulation du trafic ferroviaire).

Au vu de ces différenciations, il est important dans un premier temps d'établir les limites de notre étude. Dans le cadre de nos recherches, nous nous sommes intéressés aux systèmes distribués fournissant une information personnalisée à leurs utilisateurs avec un exemple d'application dédiée aux étudiants.

L'objectif de nos travaux est d'entrevoir la réalité d'une aide à l'étudiant par la pertinence de l'information délivrée. L'étudiant ne souhaite en effet avoir à disposition que d'informations qui l'intéressent directement.

Ce chapitre est composé de deux parties. La première partie vise à étudier les systèmes d'information sous l'angle des utilisateurs. "En quoi ces systèmes facilitent-ils la recherche et la mise à disposition d'informations pertinentes auprès des utilisateurs ? Serait-il intéressant de les améliorer et comment le faire ?" sont quelques-unes des questions que nous allons aborder. La seconde partie vise à étudier la personnalisation d'un point de vue pratique : quels sont les apports de la personnalisation dans les systèmes d'information ? , quelles sont les méthodes existantes ?, etc. Tout au long de ce chapitre, en se base sur les concepts identifiés dans [Rosé.03].

1.1. Personnalisation de l'information

La personnalisation et la qualité de l'information constituent un enjeu majeur pour l'industrie informatique. Que ce soit dans le contexte des systèmes d'information d'entreprise, du commerce électronique, de l'accès au savoir et aux connaissances ou même des loisirs, la pertinence de l'information délivrée, son intelligibilité et son adaptation aux usages et préférences des clients constituent des facteurs clés du succès ou du rejet de ces systèmes. La production massive de nouvelles ressources, conjuguée à la nature fortement hétérogène, multiforme et multilingue de l'information constituent des verrous autant technologiques que sémantiques que les systèmes d'accès et de filtrage actuels doivent lever pour produire une information pertinente et de qualité. La personnalisation a pour objectif, d'une part, de faciliter l'expression du besoin utilisateur et de rechercher des informations sur un sujet en écartant l'information non-pertinente et donc réduire considérablement l'espace de recherche et, d'autre part, de rendre cette information sélectionnée intelligible à l'usager et exploitable. La pertinence de l'information n'est cependant pas une mesure objective, généralisable à tous les utilisateurs. Elle se définit par un ensemble de critères et de préférences personnalisables spécifiques à chaque utilisateur ou communauté d'utilisateurs

1.1.1. Processus de recherche d'information

Rosé a identifiée les différents niveaux dans le processus de recherche d'information pour lesquels la personnalisation est susceptible d'apporter une valeur ajoutée.

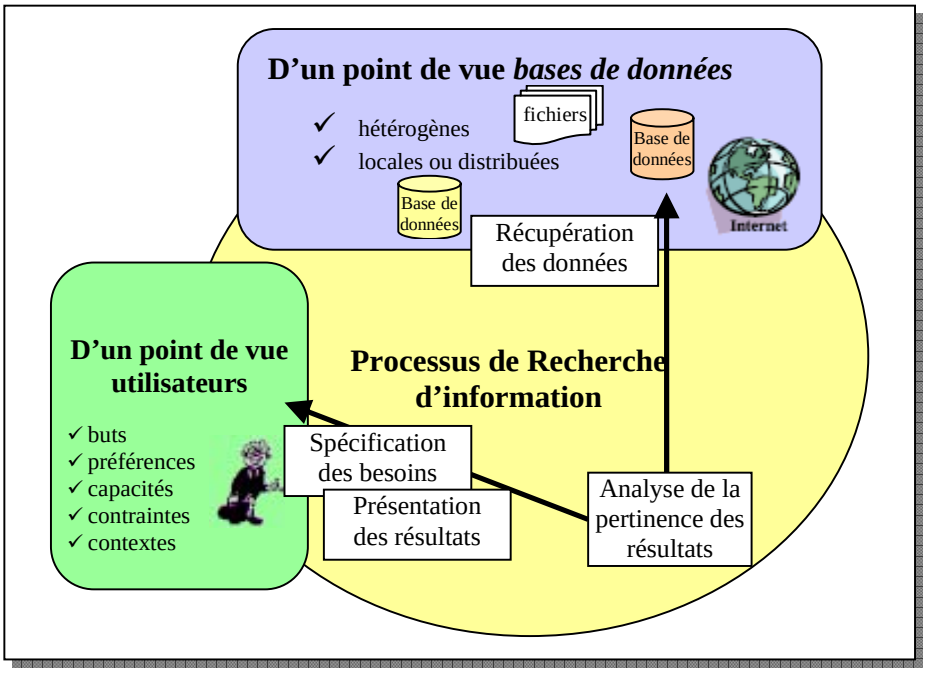


Figure 1.1 Apports de la personnalisation dans un processus de recherche d'informations [Rosé 03]

La figure 1.1 illustre un processus de recherche d'information sur lequel on montre l'intérêt de la personnalisation. La finalité d'un processus de recherche d'information est de mettre en

adéquation deux univers : l'univers de l'utilisateur avec celui des sources de données. La personnalisation vise à améliorer chaque étape du processus de recherche d'information vis-à-vis de l'utilisateur. Quatre activités importantes sont identifiées dans le cadre de systèmes d'information personnalisée (SIP) (figure 1.1) :

- la spécification des besoins,
- la récupération des données,
- l'analyse de la pertinence des résultats trouvés par rapport à l'utilisateur
- la présentation des résultats.

1.1.1.1. Spécifier les besoins

Un système d'information, personnalisée ou non, doit répondre dans la mesure du possible, à toute demande des utilisateurs. Cela consiste à prendre en considération le but que cherche à atteindre l'utilisateur. Pour un système d'information personnalisée, cela ne signifie pas seulement répondre à la requête de l'utilisateur (comme pourrait le faire un système d'information classique), mais surtout fournir uniquement les réponses pertinentes afin de satisfaire l'utilisateur.

Lors d'une requête, l'utilisateur doit formuler sa demande d'une manière compréhensible par le système d'information. La difficulté réside dans la formulation. Comment spécifier ses besoins de manière suffisamment précise pour n'obtenir que l'information recherchée mais suffisamment large pour obtenir des résultats ? La personnalisation fournit des éléments de réponse à ce problème.

L'introduction de la personnalisation dans l'étape de spécification des besoins a pour objectif de satisfaire les buts des utilisateurs *a priori* en prenant en compte le contexte de la recherche.

1.1.1.2. Récupérer les données

Une fois les besoins spécifiés, un système d'information personnalisée doit récupérer les données pertinentes. Pour cela, filtrer les solutions en fonction de l'utilisateur, c'est-à-dire en fonction de l'objectif de la requête et des préférences de l'utilisateur, peut s'avérer utile pour récupérer les données les plus pertinentes. Plusieurs techniques de filtrage existent.

Les systèmes de filtrage d'information sont des systèmes qui extraits les informations et seulement les informations pertinentes pour l'utilisateur. Ces systèmes appuient leur recherche sur des profils utilisateurs. Le profil permet de dégager des règles de filtrage, puis est mis à jour grâce aux retours d'information donnés par les utilisateurs [Aas97]. Il existe deux types de méthodes pour filtrer des informations : les méthodes dites cognitives et les méthodes dites sociales.

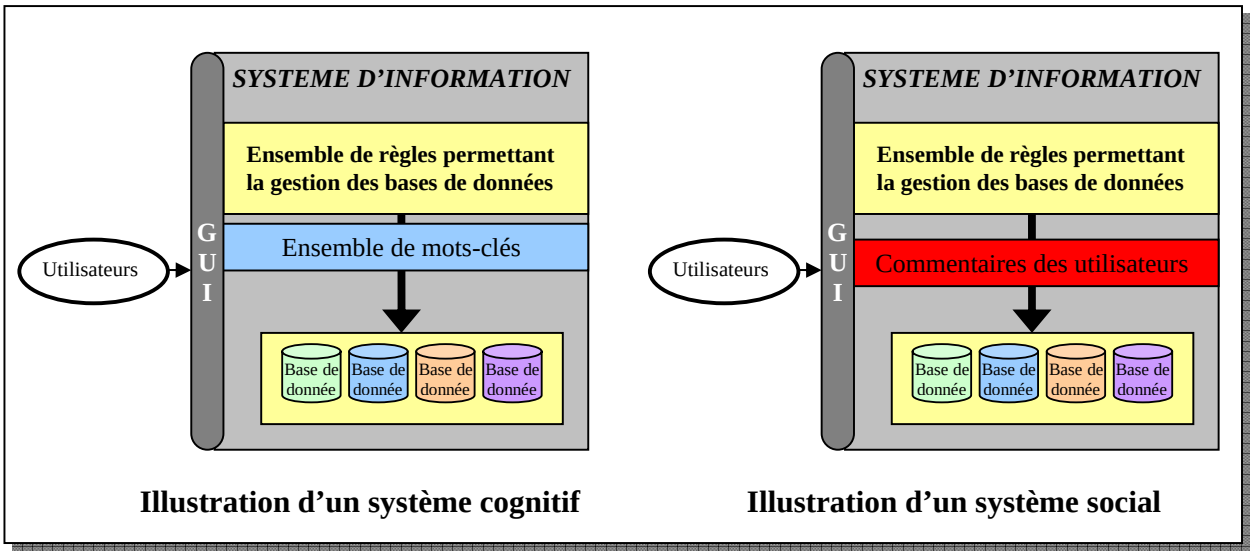


Figure 1.2 Représentation schématique d'un système d'information s'adaptant aux buts des utilisateurs [Rosé 03]

Méthode cognitive :

Cette méthode se base sur le calcul vectoriel. Le profil de l'utilisateur est représenté par un vecteur de mots-clés ou de poids associés à ces mots-clés. Tous les documents sont représentés de la même manière. La méthode consiste à calculer la distance entre le vecteur représentatif de l'utilisateur et ceux des documents. La distance la plus faible correspond au document le plus pertinent pour la requête de l'utilisateur.

L'analyse s'effectue de la manière suivante [Rosé .03]:

- le système extrait des phrases significatives de chaque document jugé intéressant par l'utilisateur
- un arbre de décision est créé à partir de ces phrases
- le système transforme chaque arbre en une requête booléenne ;
- cette requête est soumise à un moteur de recherche afin de trouver de nouveaux documents. Cependant, cette approche présente des inconvénients :
- cette méthode ne peut évaluer que des documents sous forme textuelle. De ce fait, les nouveaux types de documents (vidéo, photographie, son, etc.) ne peuvent pas être gérés par ces systèmes [Shardanand et al.95]
- cette méthode est incapable de refléter les changements d'intérêts des utilisateurs.

Méthode sociale :

Comme nous avons l'habitude de le faire dans la vie courante pour se faire une opinion sur un film, un site web, ou autre que nous n'avons pas encore vus, nous profitons de l'expérience d'amis dont les préférences sont proches des nôtres et qui ont déjà évalué un film ou une page web. Nous construisons donc notre opinion en estimant ce que serait notre évaluation en fonction des évaluations des autres. Cette estimation, permet donc de filtrer

les ressources potentiellement pertinentes pour l'utilisateur. Ceci constitue le fondement du filtrage collaboratif.

Cette méthode peut être appliqué sur tous types d'informations. Cependant, elle implique fortement l'utilisateur (ses préférences et contraintes). En effet, Ce dernier doit valider ces choix et son historique sur lesquelles se base un tel système.

1.1.1.3. Générer une solution pour l'utilisateur

Cette étape constitue la phase d'évaluation et de comparaison entre le but initial et la pertinence des résultats trouvé par le système. En fait, il est important de vérifier la compatibilité de la solution avec les buts de l'utilisateur a posteriori.

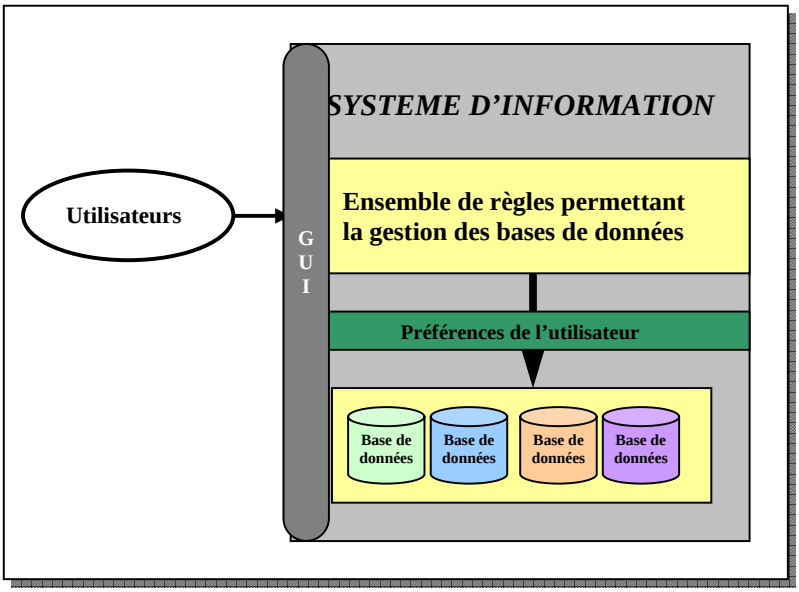


Figure 1.3 Représentation schématique d'un SI s'adaptant aux préférences de l'utilisateur [Rosé .03]

1.1.1.4. Présenter les résultats

La phase de présentation des résultats est une phase importante pour parler de la personnalisation de l'information globale : du point de vue contenant et de point de vue contenu. En effet, cette étape consiste à présenter les informations selon les préférences de l'utilisateur [Rosé .03]. L'adaptation des informations en fonction d'un ensemble de préférences est un des thèmes de recherche de la communauté des chercheurs en Interaction Homme-Machine (IHM) [Norcio et al.89] dans le cadre des interfaces adaptatives.

En outre, la personnalisation de la présentation des données est en relation forte avec les supports avec lesquels l'utilisateur interroge. En effet, la surface de l'interface d'un téléphone portable n'est pas la même que celle d'un ordinateur. La quantité d'information diffère. Pourtant l'essentiel des informations doit apparaître sur n'importe quelle interface. C'est pourquoi il est nécessaire d'adapter la quantité et la forme des informations, la navigation dans l'interface, le placement des objets graphiques, etc., selon la cible visée. [Rosé .03]

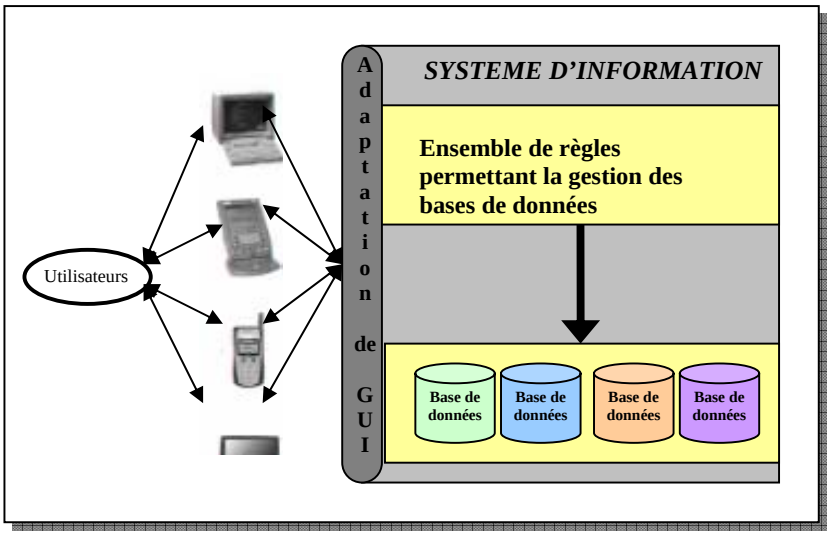


Figure 1.4 Représentation schématique de la plasticité d'un SI [Rosé .03]

1.1.2. Système d'information personnalisé

Cette section a présenté ce qui est attendu par un système d'information personnalisée. Actuellement, certains systèmes proposent d'adapter les informations selon le but de l'utilisateur (par filtrage), selon les capacités de l'utilisateur (notamment le support avec lequel il interroge le système) ou selon ses préférences (par le biais d'un profil utilisateur). Pourtant ces adaptations sont complémentaires. Un système d'information personnalisée devrait adapter les données selon le but, les préférences et les capacités de l'utilisateur. La figure 1.5 illustre ce concept. La complémentarité des techniques et modèles devrait dans ce cadre améliorer les performances de l'ensemble.

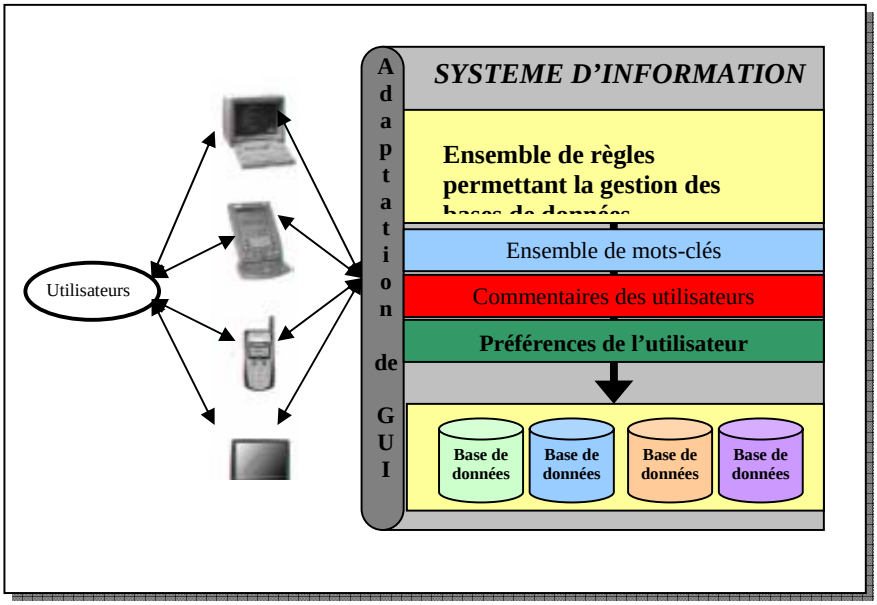


Figure 1.5 Représentation schématique d'un système d'information personnalisé

Les apports de la personnalisation de l'information se voient importantes et utiles pour les systèmes d'information actuels, notamment devant la grande masse de données ainsi, que son hétérogénéité. Cependant, la personnalisation de l'information se voit face à plusieurs problèmes. Nous intéressons dans nos recherches au problème d'intégration de sources de données hétérogène, sui sera détaillé dans la section suivante.

1.2. Problématique : Intégration de données hétérogènes et reparties

L'accès à une information pertinente, adaptée aux besoins et au contexte de l'utilisateur est un challenge dans un environnement Internet ou GRID, caractérisé par une prolifération des ressources hétérogènes (données structurées, documents textuels, composants logiciels, images), conduisant à des volumes considérables. Au fur et à mesure que ce volume s'accroît et que les données se diversifient, les systèmes de recherche d'informations (moteurs web, SGBD, etc.) délivrent des résultats massifs en réponse aux requêtes des utilisateurs, générant ainsi une surcharge informationnelle dans laquelle il est souvent difficile de distinguer l'information pertinente d'une information secondaire ou même du bruit. Que ce soit dans le contexte des systèmes d'information d'entreprise, du commerce électronique, de l'accès au savoir et aux connaissances ou même des loisirs, la pertinence de l'information délivrée, son intelligibilité et son adaptation aux usages et préférences des clients constituent des facteurs clés du succès ou du rejet de ces systèmes.

1.2.1. Systèmes d'information hétérogènes

Etant conçus par des communautés différentes, les systèmes d'information sont autonomes et hétérogènes. L'hétérogénéité se situe à deux niveaux : syntaxique et sémantique.

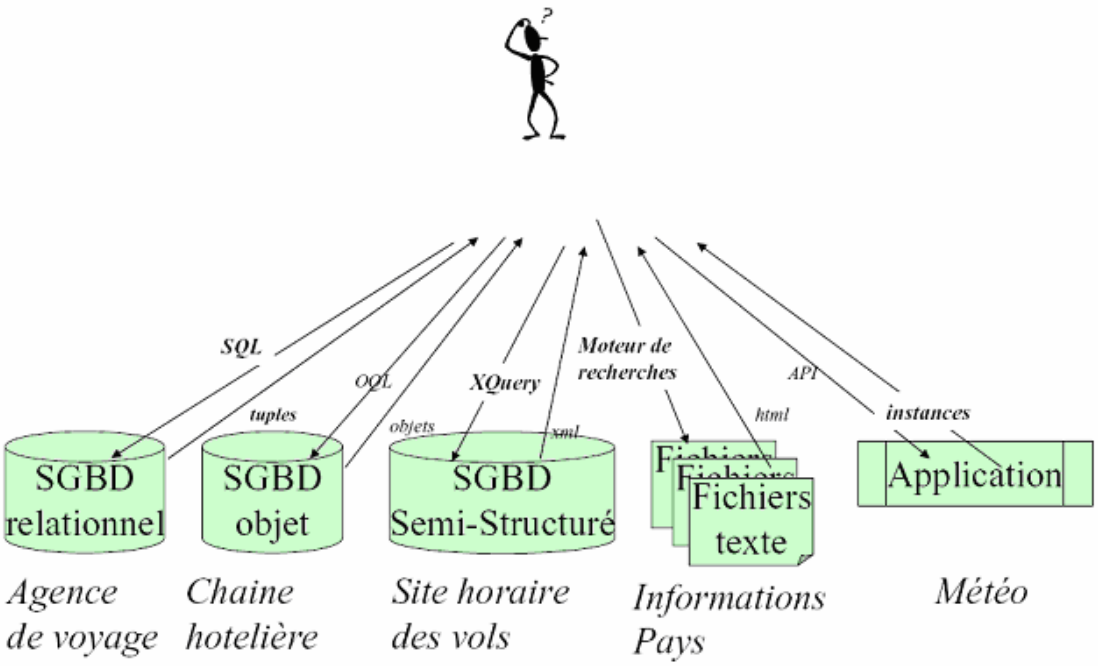


Figure 1.6 Exemple schématique des hétérogénéités des sources de données.

- **L'hétérogénéité syntaxique** : se retrouve dans les formats de stockage des données (XML, relationnel, objet, etc.), dans les langages d'interrogation (XQuery, SQL, OQL, etc.), dans les protocoles d'accès (HTTP, etc.), dans les interfaces, etc.
- **L'hétérogénéité sémantique** : représente les différences entre les interprétations du monde réel selon le contexte et l'utilisation des données. Ceci a généralement lieu durant le processus de conception des systèmes d'information. Les conflits sémantiques peuvent survenir au niveau des schémas et des données [Park et Ram.04].

Les conflits au niveau des données résultent de l'utilisation de domaines de données différents selon le type d'application (e-gov, e-commerce, bioinformatique, etc.). En effet, des données similaires peuvent être représentées et interprétées différemment dans chaque domaine.

Les conflits des schémas sont, quant à eux, caractérisés par des différences dans les structures logiques ou méta données.

Dans ce contexte, Goh [Goh, Bressan et al, 99] a identifié quatre principaux types de conflit: conflits de nom, conflits d'échelle, conflits d'indéterminisme (*confounding conflicts*) et conflits de représentation. On les commente dans ce qui suit.

- Les conflits de nom sont liés aux différences dans la désignation de concepts. Le cas le plus fréquent est la présence de **synonymes**, d'**homonymes**, d'**hyperonymes** et d'**hyponymes** [Mena, Illarramendi et al., 96]. Les **synonymes** sont deux mots distincts ayant le même sens. C'est l'exemple de "publication" et "article" qui capturent la même information sur les articles de recherche publiés. Les **homonymes** sont des mots partageant la même graphie et la même prononciation mais n'ayant pas le même sens. Par exemple, "mémoire" peut faire référence à des entités différentes: "mémoire informatique" et "mémoire de thèse". Les **hyperonymes** et les **hyponymes** indiquent des niveaux d'une hiérarchie désignés par le concept plus "général" ou le concept moins "général". C'est le cas de "personne" qui est un hyperonyme de "employé" car c'est un terme plus "général".
- Les conflits d'échelle ont lieu quand des systèmes de référence différents sont utilisés pour mesurer une grandeur. C'est l'exemple de "Fahrenheit" ou "Celsius" pour la température.
- Les conflits d'indéterminisme surgissent quand les concepts semblent avoir le même sens alors qu'ils sont différents. Ceci peut être dû à des contextes temporels différents.
- Les conflits de représentation surviennent quand les schémas de deux sources décrivent différemment un même concept. Par exemple, le nom d'un étudiant peut être représenté par deux champs "prénom" et "nom" dans une source et par un seul champ "identité" dans une autre source. Nous pouvons également citer l'exemple d'un concept défini comme une classe dans une source de données et comme attribut dans une autre, etc.

Masquer l'hétérogénéité sémantique revient à faire communiquer les systèmes d'information via une connaissance commune qui permet d'expliciter et de préciser le sens des données pour être interprétées correctement par différents systèmes.

1.2.2. Intégration de l'information

Les systèmes d'intégration de l'information fournissent une vue unifiée de multiples systèmes hétérogènes, autonomes et répartis, facilitant ainsi l'accès à l'information. Ceci est réalisé par l'utilisation d'un schéma global, qui fournit une vue réconciliée (consensuelle) des sources locales. Il existe deux approches pour l'intégration : l'intégration physique (Matérialisée) et l'intégration virtuelle. La première consiste à créer un entrepôt de données à partir des sources locales, dupliquant ainsi les données (voir Figure 1.7.). L'intégration physique a l'avantage de fournir des temps d'accès rapides mais nécessite un support de stockage volumineux et fiable et des outils spécifiques, appelés ETL (*Extract, Transform and Load*), pour le traitement préalable des données

La solution de l'intégration virtuelle, que nous avons adoptée, permet d'avoir une vue *fraîche* des données, sans avoir à les dupliquer ni à les transformer. L'intégration virtuelle de sources hétérogènes et autonomes est fondée sur la médiation [Wiederhold,92] (voir Figure 1.8.). Cette dernière repose sur deux niveaux : le médiateur et les adaptateurs. Le médiateur a pour fonction d'offrir une vue unifiée des différentes sources de données grâce au schéma global, cachant en cela leur hétérogénéité et leur répartition. Il offre un protocole d'accès et un langage de requêtes commun à toutes ces sources. Quant à l'adaptateur, il adapte la requête exprimée dans le langage commun au langage de la source, tout en utilisant le bon protocole d'accès. Etant donné que la requête utilisateur est exprimée en fonction du schéma global, alors qu'elle doit être exécutée par les sources locales, un mapping ou correspondance entre ce schéma global et les schémas locaux (des sources) est nécessaire. Ce mapping constitue un traitement clé dans le processus général. Il sera utilisé pour réécrire la requête initialement exprimée en fonction du schéma global, en des sous-requêtes exprimées, chacune, en fonction du schéma local de la source qui l'exécutera.

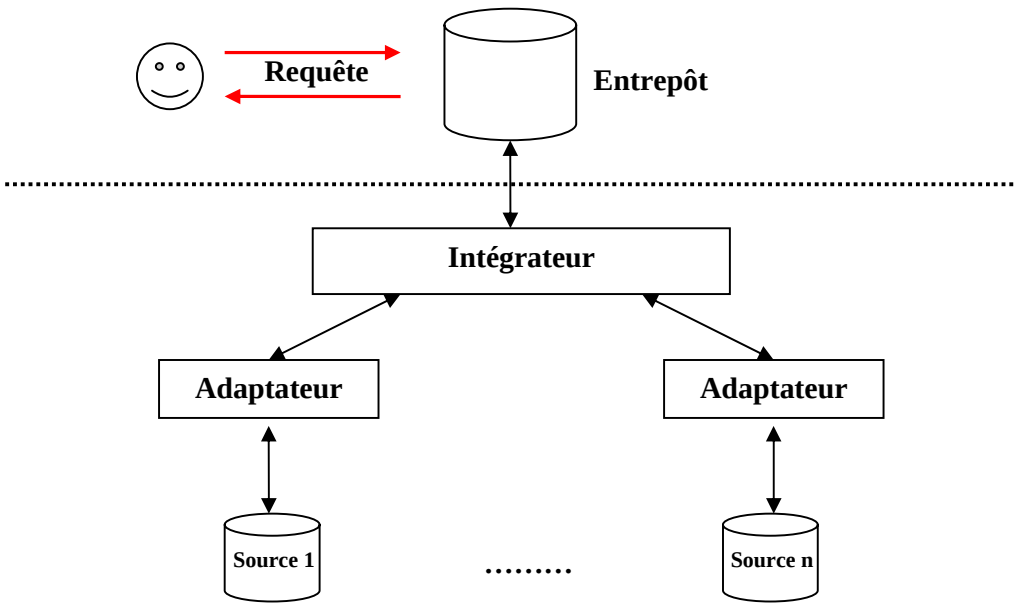


Figure 1.7 Intégration physique des sources de données hétérogènes.

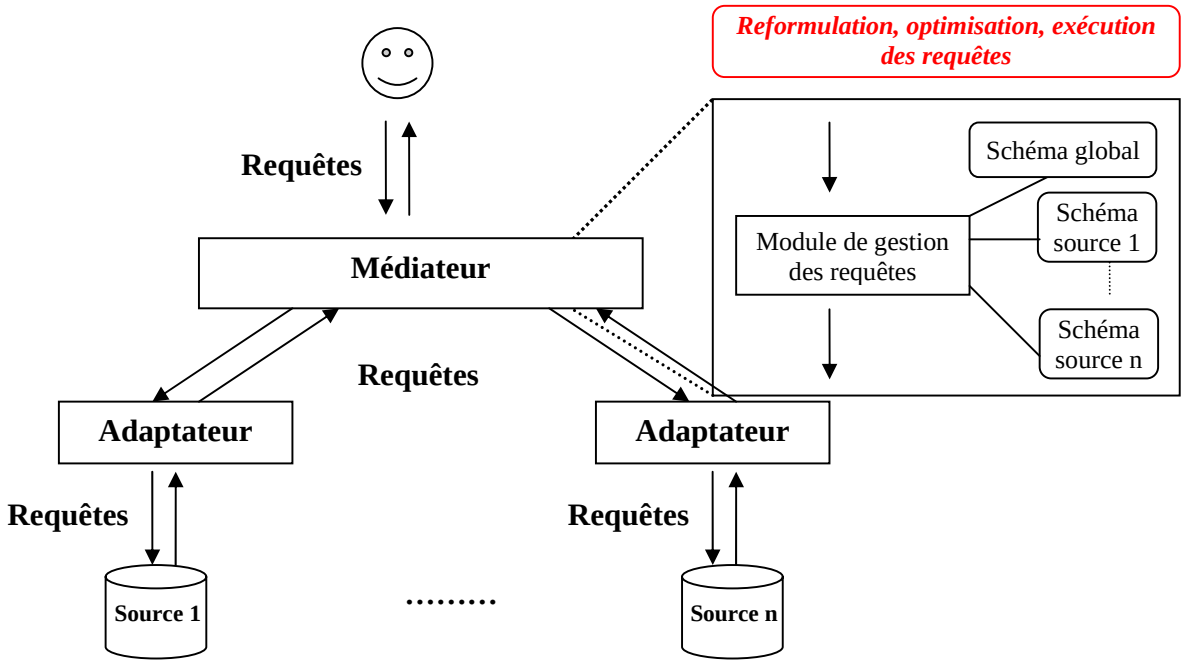


Figure 1.8. Intégration virtuelle des sources de données hétérogènes

Deux approches existent pour définir le mapping entre le schéma global et les schémas des sources: *Local As View* (LAV) et *Global As View* (GAV) [Halevy, 01] (ces deux approches seront décrites par la suite).

Comparaison entre l'approche physique et l'approche virtuelle

On donne un tableau récapitulatif qui compare l'approche matérialisée et l'approche virtuelle :

	Approche physique	Approche virtuelle
Performances	+	-
Historiques	+	-
Volume	-	+
Actualisation temps réel	-	+
Ajout/suppression de sources	-	+
Complexité	-	+

Approche virtuelle préférable si :

- les sources sont mises à jour fréquemment
- les sources sont très nombreuses
- il est impossible de prédire les requêtes de l'utilisateur

1.2.3. Les Systèmes de médiation [Bouzeghoub.06]

Les systèmes d'intégration de données offrent des architectures d'interopérabilité sur une fédération de sources de données distribuées, autonomes et hétérogènes (voir Figure 1.9.).

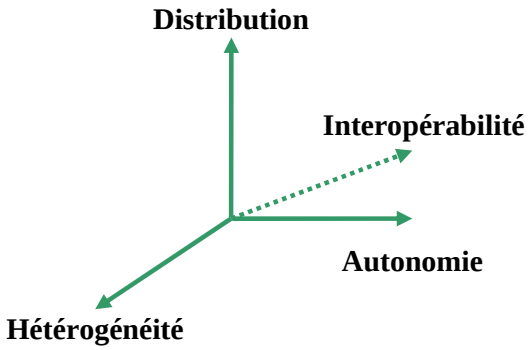


Figure 1.9. Caractéristiques des sources de données hétérogènes.

Les entrepôts de données, les systèmes de médiation et les architectures P2P sont des exemples d'infrastructures qui permettent l'intégration de données, c'est-à-dire l'accès à des données produites par des sources autonomes. A travers des schémas virtuels, des méta-données et des correspondances sémantiques, ils permettent d'accéder à ces sources de données de façon uniforme et transparente, en transformant par réécriture les requêtes d'un utilisateur en sous-requêtes envoyées aux sources de données les plus appropriées. L'hétérogénéité des données extraites des sources nécessite leur réconciliation (transformation, nettoyage), c'est-à-dire leur mise en conformité par rapport au schéma du médiateur, avant de les délivrer à l'utilisateur.

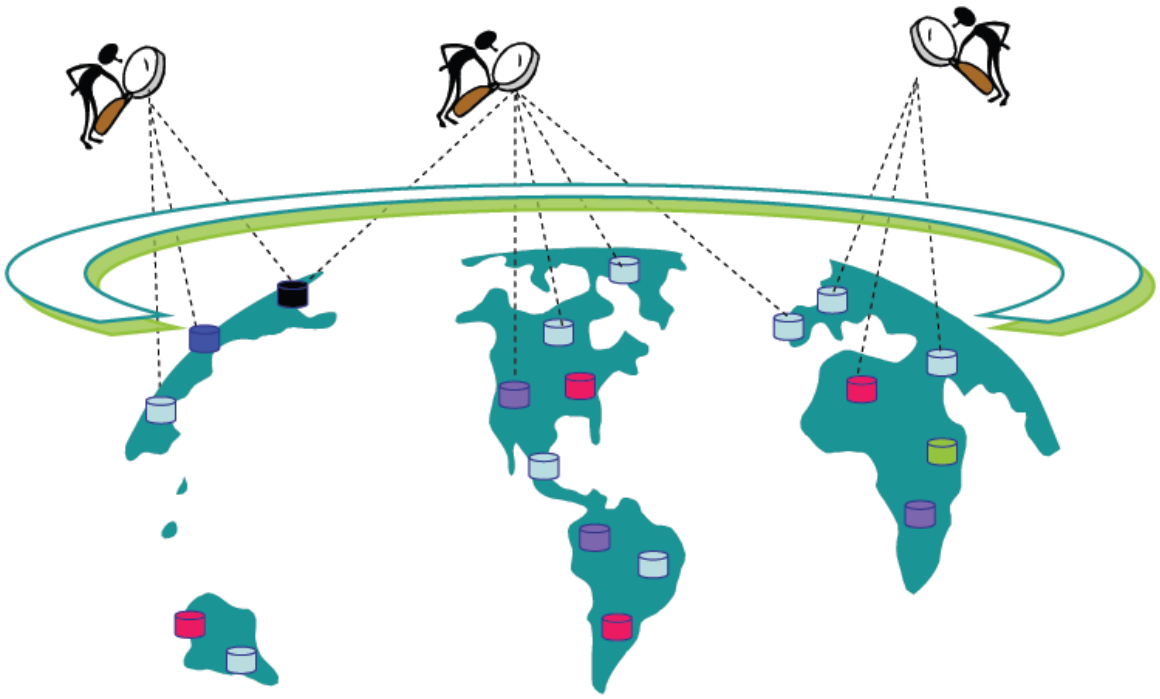


Figure 1.10. Accès transparent aux données hétérogènes.

Dans les systèmes de médiation, les besoins des utilisateurs sont représentés par un schéma de médiation pouvant être créé à partir des schémas des sources de données, ou indépendamment des sources par des experts du domaine. Il existe essentiellement deux types de liens entre le schéma de médiation et les schémas des sources de données : les correspondances sémantiques et les requêtes de médiation. Une correspondance sémantique met en relation deux concepts équivalents. Les requêtes de médiation décrivent le mode de calcul des instances du schéma de médiation à partir des instances des schémas sources ou inversement. Ces requêtes peuvent être élaborées suivant deux approches : dans la première, dite global-as-view (GAV), chaque objet du schéma de médiation est défini par une requête exprimée sur les données sources. Dans la deuxième approche, dite local-as-view (LAV), chaque objet dans un schéma source est défini par une requête exprimée sur le schéma de médiation. Dans les deux cas, les requêtes des utilisateurs sont exprimées sur le schéma de médiation et des algorithmes de réécriture permettent de transformer ces requêtes sur les données sources.

L'approche GAV est souvent préférée pour sa simplicité d'implémentation, mais la définition des requêtes de médiation est généralement faite manuellement, tâche très fastidieuse et très difficile car elle nécessite non seulement une connaissance approfondie du contenu de toutes les sources de données, mais également des correspondances sémantiques entre ces sources (lorsqu'elles existent) et du schéma de médiation. La complexité de cette tâche s'accroît si le nombre de sources de données est important.

Nous nous sommes intéressés à trois problèmes complémentaires: la génération automatique de requêtes de médiation en tenant compte de l'hétérogénéité des données, l'évolution de ces requêtes lorsque les besoins ou les sources de données changent et la découverte au sein d'une même source, lorsqu'elles ne sont pas explicites, de contraintes d'intégrité entre les données (dépendances fonctionnelles, dépendances d'inclusion).

1. *La génération automatique de requêtes de médiation* est un problème difficile pour deux raisons: (i) soit on fournit au concepteur des outils d'exploration de métadonnées, charge à lui de décider quelle source et quelle information l'intéresse, auquel cas la génération ultime de requêtes de médiation devient alors triviale ; mais cette exploration peut être fastidieuse et se perdre dans un processus de navigation interminable lorsque les sources de données sont nombreuses ; (ii) soit on tente d'exploiter automatiquement ces métadonnées pour générer tous les mappings possibles entre un schéma de médiation et ses sources, avec le risque de proposer plusieurs sémantiques différentes à l'utilisateur et de faire face à des algorithmes combinatoires. La première approche a été celle du projet Clio d'IBM et de l'Université de Toronto [MHHe00][VMPo03]. Ces approches considèrent toutes une source de données unique et ne tiennent pas compte de l'hétérogénéité des données. Plusieurs approches ont été proposées pour traiter du problème d'hétérogénéité des données ; certaines proposent une méthodologie de génération de règles de conversion de valeurs numériques, basée sur des techniques statistiques [CFLM01], d'autres proposent un ensemble d'opérateurs servant de briques de base à la réalisation manuelle de programmes de nettoyage [GFSS+01] ; d'autres encore préconisent la définition d'un langage conceptuel basé sur les logiques de description pour exprimer les transformations comme des ornements de requêtes de médiation

(*adorned queries*) [CGLN+99]. Nos travaux se situent dans un contexte similaire à ce dernier dans la mesure où nous nous intéressons à la validité opérationnelle des requêtes de médiation.

2. *L'évolution des requêtes de médiation* est l'un des deux principaux verrous, avec l'explosion combinatoire, dans la génération des requêtes de médiation. Quelle que soit l'approche utilisée (définition interactive ou génération automatique), une fois les requêtes de médiation générées, elles peuvent devenir aussitôt obsolètes par la disparition de certaines sources ou la modification des schémas des autres. Dans un contexte de sources de données autonomes et à grande échelle, ces changements ne sont pas forcément rares. Le processus de génération devient alors intimement lié à celui de l'évolution. Ce problème de l'évolution, souvent appelé synchronisation de vues [LNRu02], n'a été pris en compte que récemment et très partiellement dans les projets : [LNRu02] aborde l'évolution des requêtes de médiation uniquement sous l'angle de la suppression des sources et du renommage de leurs constituants, [McPo02] propose une approche d'annotation des requêtes de médiation pour indiquer quelle type d'évolution est acceptée, [VMPo03] s'intéresse à la mesure de la distance sémantique entre la nouvelle requête issue de l'évolution et la requête ancienne. Notre approche du problème, basée sur des règles actives, est similaire à la propagation des mises à jours à travers les vues matérialisées.

1.2.4. Approche de médiation personnalisée

Nous décrivons à ce niveau le principe d'une approche de personnalisation de l'information dans un contexte de médiation à grande échelle où les sources de données sont évolutives et sujettes à des déconnexions temporaires ou permanentes.

Dans ce contexte, l'évaluation d'une requête se fait en tenant compte, d'une part, du profil utilisateur qui enrichira son expression, et, d'autre part, des sources disponibles et de leur qualité au moment de l'évaluation de la requête. Nous présenterons en particulier deux approches de personnalisation:

- une approche enrichissement-réécriture (ER) qui enrichit d'abord la requête de l'utilisateur à l'aide du profil de ce dernier avant de la réécrire sur les sources de données;

- une approche réécriture-enrichissement (RE) qui effectue d'abord la réécriture de la requête utilisateur et introduit ensuite des enrichissements sur les réécritures résultantes.

1.2.4.1. Approche Enrichissement-Réécriture

Le premier scénario consiste à enrichir d'abord la requête utilisateur à l'aide de son profil, sans tenir compte des sources et ensuite à rechercher les réécritures possibles de la requête enrichie. L'objectif visé par cette approche est de prendre en compte les préférences les plus importantes pour l'utilisateur. En effet, étant donné que l'enrichissement dans ce cas est fait sur la requête initiale, on y ajoute les Top K prédicats non contradictoires avec elle. On obtient ainsi l'enrichissement le plus pertinent en fonction du profil utilisateur et de la requête initiale. On peut formaliser cette approche comme suit :

$$R(\varepsilon(Q_u /[P_u, S_v]) / S_m)$$

$\varepsilon(Q_u/[P_u, S_v])$ délivre l'enrichissement Q_{u+} de la requête Q_u par rapport au profil utilisateur P_u et au schéma virtuel S_v. $R(Q_{u+}/ S_m)$ est la réécriture de la requête enrichie Q_{u+} par rapport à l'ensemble S_m des définitions LAV des sources de données.

Nous avons vu que l'algorithme d'enrichissement peut ajouter de nouvelles relations virtuelles à la requête initiale, ce qui se traduit par la construction de nouveaux tas de sources contributives pendant la phase de réécriture (un nouveau tas pour chaque nouvelle relation virtuelle). Comme la recherche des réécritures candidates est un problème combinatoire, chaque nouveau tas qui contient plus d'une source contributive multiplie le nombre de réécritures candidates à explorer par la cardinalité de ce tas. Cependant, le nombre de réécritures candidates obtenues par l'approche ER n'est pas toujours supérieur à celui du scénario R-E. Le fait d'ajouter des prédicats supplémentaires à la requête utilisateur sans prendre en compte les possibilités de réponse des sources permet d'augmenter la pertinence des résultats obtenus, mais peut éliminer des sources contributives pour sa réécriture.

Dans certains cas il est possible que la requête enrichie ne puisse pas être réécrite parce qu'il n'y a pas de sources contributives pour une des relations virtuelles. Une solution possible à ce problème est de prendre en compte les contraintes sur le contenu des sources avant de choisir les prédicats pour la phase d'enrichissement. Dans ce cas on inverse l'ordre de l'application des deux algorithmes ce qui revient à enrichir les réécritures possibles de la requête initiale. La section suivante décrit cette approche.

1.2.4.2. Approche Réécriture-Enrichissement

L'objectif de cette approche est de s'assurer que chaque prédicat qui est ajouté à la requête de l'utilisateur pendant la phase d'enrichissement est exécutable i.e. il y a potentiellement des résultats qui le satisfont. L'idée principale est d'effectuer la réécriture de la requête en premier, ce qui permet d'obtenir plusieurs sous-requêtes (réécritures) qui contiennent les prédicats de sélection des sources qu'elles interrogent en plus des prédicats de la requête utilisateur. Ensuite, chacune des réécritures est enrichie à l'aide du profil utilisateur et des profils des sources correspondantes. Cette approche peut être formalisée par l'expression suivante.

$$\varepsilon(R(Q_u / S_m)/[P_u, S_m])$$

$R(Q_u/ S_m)$ l'ensemble des réécritures des w_u de la requête Q_u par rapport à l'ensemble S_m des définitions LAV des sources de données. $\varepsilon(Q_u/[P_u, S_v])$ délivre les enrichissements de l'ensemble w_u de réécritures de Q_u avec le profil P_u.

Un premier constat que nous pouvons faire est que les prédicats qui sont utilisés pour l'enrichissement d'une réécriture candidate peuvent être moins pertinents que ceux utilisés pour enrichir la requête initiale. Si un ou plusieurs prédicats du profil, parmi les Top K qui ne sont pas contradictoires avec la requête utilisateur, sont en contradiction avec les prédicats des sources, ils seront remplacés par les prédicats qui les suivent dans le profil (à condition d'avoir plus de prédicats dans Pu). Les nouveaux prédicats ont un poids au plus égal à celui du prédicat le moins « intéressant » parmi le Top K initial. Autrement dit, on choisit des prédicats dont la position dans le profil utilisateur est plus basse par rapport au dernier prédicat qui aurait été choisi avant la réécriture. D'un autre coté, si au lieu d'être contradictoires avec la réécriture, les prédicats sont remplacé parce qu'ils sont satisfaits par la définition des sources utilisées, on va obtenir au final des requêtes plus pertinentes parce qu'elles vérifient plus de prédicats du profil utilisateur.

Le risque que crée la prise en compte des contraintes des sources avant l'enrichissement est que le profil ne puisse pas être pris en compte. Si la définition des sources est trop riche (i.e. contient beaucoup de prédicats) et que par conséquent, après réécriture, tous les prédicats du profil sont soit contradictoires soit satisfaits par les sources utilisées, alors l'enrichissement n'apporte aucune information supplémentaire. Le résultats obtenu est le même que si on applique uniquement la réécriture. Or l'objectif de l'accès personnalisé est de cibler mieux les besoins de l'utilisateur.

Un problème supplémentaire vient du fait que les sources utilisées pour les différentes réécritures candidates n'ont pas les mêmes contraintes sur leur contenu et de ce fait il se peut qu'elles ne soient pas enrichies avec les mêmes prédicats. Par conséquent, les requêtes reformulées finales n'auront pas la même pertinence.

Bien que cette approche permette d'obtenir des enrichissements exécutables, elle a un inconvénient majeur qui vient du fait que la réécriture de la requête initiale fixe le schéma des relations qu'elle interroge. En fait, l'algorithme d'enrichissement peut ajouter de nouvelles relations à la requête initiale s'il y a des prédicats qui sont exprimés sur leurs attributs.

Lorsque la réécriture est faite en premier, ceci n'est plus possible et seuls les prédicats exprimés sur les attributs, présents dans les schémas des sources utilisées pour la réécriture de la requête initiale, peuvent être ajoutés.

1.2.4.3. Comparaison des deux approches : R-E et E-R

De façon générale, l'approche E-R permet de prendre en compte un plus grand nombre de préférences du profil utilisateur. Lorsque l'enrichissement de la requête initiale est fait en premier, tous les prédicats du profil sont potentiellement utilisables i.e. sont exprimés sur des attributs qui sont présents dans le schéma sur lequel est exprimée la requête. Si un prédicat porte sur un attribut qui appartient à une relation non présente dans la requête, cette relation y est ajoutée à l'aide des prédicats de jointure. Cette extension de la portée de la requête est impossible dans l'approche R-E où l'enrichissement est fait sur un schéma fixe qui est celui des relations sources choisies. Ceci implique que les préférences exprimées sur des attributs non présents dans ce schéma ne peuvent pas être utilisées.

Pire encore, supposons que pour chaque réécriture candidate w_u^i de la requête utilisateur Q_u et chaque prédicat p_j du profil utilisateur Pu, une des trois conditions suivantes soit vérifiée :

- p_j est satisfait par la définition des sources utilisées pour la réécriture w_u^i
- p_j est contradictoire avec la définition des sources de w_u^i
- p_j est exprimé sur un attribut qui n'apparaît pas dans le schéma des sources de w_u^i

Dans ce cas, aucun prédicat du profil ne peut être utilisé pour l'enrichissement des réécritures candidates. Par conséquent l'approche R-E ne personnalise pas les résultats contrairement au scénario E-R qui permet de prendre en compte les prédicats qui vérifient une des deux dernières conditions.

Dans l'approche E-R, ajouter à la requête utilisateur des prédicats contradictoires avec les définitions des sources, permet de réduire le nombre de sources contributives à la réécriture de

celle-ci. L'inconvénient de cette propriété est le risque d'obtenir une requête qui ne peut pas être réécrite (si on élimine l'ensemble des sources contributives pour un but). Par contre si le nombre de sources disponibles est trop important et si les prédicats de la requête ne sont pas très restrictifs, la réduction du nombre de sources à prendre en compte peut être bénéfique, constituant ainsi un filtre supplémentaire qui réduit la quantité des résultats.

De façon générale, l'approche E-R permet de prendre en compte les préférences les plus importantes du profil utilisateur. Cependant si les conditions suivantes sont vérifiées, l'approche R-E génère des requêtes qui vérifient d'avantage de prédicats du profil :

- tous les Top K prédicats choisis pour l'enrichissement de la requête initiale Qu sont exprimés sur des attributs présents dans le schéma des sources qui peuvent être utilisées pour sa réécriture,
- aucun prédicat du Top K initial n'est contradictoire avec la définition des sources des réécritures candidates,
- au moins un prédicat du Top K initial est satisfait par les définitions des sources qui sont choisies pour les réécritures candidates,
- parmi les prédicats du profil en dehors du Top K initial, il y a certains qui peuvent être utilisés pour l'enrichissement des réécritures Wu de Qu (ne sont pas contradictoires avec les définitions des sources et sont exprimés sur des attributs présents dans leurs schémas).

Dans cette situation, l'enrichissement des réécritures candidates obtenu par l'approche R-E, est fait avec les prédicats du Top K initial qui ne sont pas satisfaits par les définitions des sources choisies, complétés avec les meilleures préférences du profil parmi celles qui restent et qui peuvent être utilisées. Il est important de remarquer que les requêtes obtenues après enrichissement prennent en compte l'ensemble des prédicats du Top K initial parce que les prédicats remplacés sont satisfaits par la définition des sources. Les résultats de l'approche R-E vérifient un plus grand nombre de préférences par rapport à ceux du scénario E-R et on peut considérer qu'ils sont plus pertinents pour l'utilisateur. Un cas extrême est que l'ensemble des prédicats du Top K initial soient satisfaits par la définition de l'ensemble des réécritures candidates ce qui implique que l'approche E-R n'apporte aucune personnalisation des résultats par rapport à une exécution classique de la requête.

Bien que les objectifs et le fonctionnement des deux approches soient différents, il se peut qu'elles produisent les mêmes résultats. Ceci est possible si l'ensemble des Top K prédicats choisis pour l'enrichissement est le même dans les deux approches et si les réécritures candidates sont faites avec les mêmes combinaisons de requêtes de médiation. Une telle situation se produit si aucun prédicat du Top K initial ne vérifie une des trois conditions citées plus haut. Dans ce cas le choix des sources contributives pour la réécriture ne dépend que des prédicats de la requête initiale et les préférences du Top K initial peuvent être utilisés pour la phase d'enrichissement quelque soit l'approche de reformulation (E-R ou R-E) de médiation. Une telle situation se produit si aucun prédicat du Top K initial ne vérifie une des trois conditions citées plus haut. Dans ce cas le choix des sources contributives pour la réécriture ne dépend que des prédicats de la requête initiale et les préférences du Top K initial peuvent être utilisés pour la phase d'enrichissement quelque soit l'approche de reformulation (E-R ou R-E).

Un deuxième cas d'égalité des deux approches apparaît si : (i) les prédicats obligatoires de l'enrichissement ainsi que les combinaisons des requêtes de médiation de la réécriture sont les mêmes dans les deux approches et (ii) il n'y a pas d'autres prédicats du profil qui peuvent remplacer ceux qui sont contradictoires ou satisfaits par les définitions des sources dans le scénario R-E. L'hypothèse de l'égalité des prédicats obligatoires garantie que les résultats des deux approches vérifient les mêmes prédicats.

Dans le cas contraire, si un prédicat obligatoire est satisfait par les requêtes de médiation d'une réécriture candidate, il sera remplacé par un des prédicats optionnels ce qui fait que les résultats obtenus par la requête enrichie vont vérifier au minimum M+L+1 prédicats au lieu des M+L initialement.

En résumé, l'approche E-R est orienté vers la satisfaction des préférences les plus importantes pour l'utilisateur et permet de prendre en compte plus de prédicats de son profil, excepté dans les cas particuliers cités dans cette section. Alors que l'approche R-E permet de garantir l'exécutabilité des requêtes obtenues après reformulation.

1.3. Méthodologie de développement d'un SIP « PerMet »

1.3.1. Présentation globale de la méthode

Afin de développer SIP (figure 1.11), qui réutilise les acquis lors de la personnalisation du SI une démarche méthodologique consiste à distinguer le SI de la personnalisation elle-même. En effet la séparation entre le SI et le SP permet l'intégration des SP sur des SI existants. [Anli 01]

Ainsi, cette méthode permet aussi bien la mise en place d'un nouveau système d'information personnalisé que la personnalisation d'un système d'information déjà existant en prenant en compte différentes modalités d'entrées-sorties, différents canaux de communication et différentes plate-forme d'interaction.

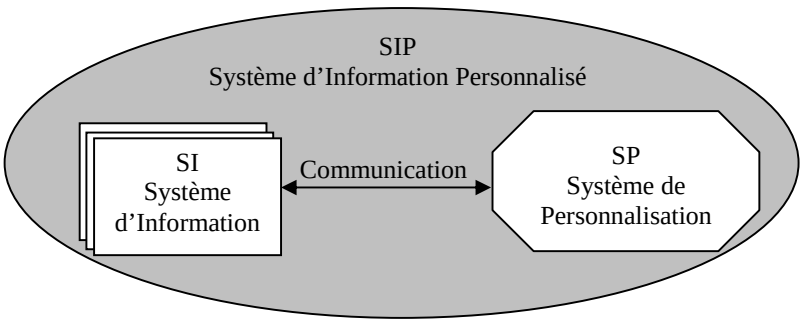


Figure 1.11. Vue générale d'un système d'information personnalisé (SIP).

La méthode que nous présentons ici suit un modèle de développement suivant trois parties (voir Figure 1.11.). La partie SI concerne le développement d'un service de SI. La partie SP concerne l'adaptation et la configuration d'un SP. Ces deux parties SI et SP suivent un modèle de développement classique pouvant se dérouler en parallèle et se rejoignent pour former la partie du milieu. Ce modèle développement est itératif et incrémental. La réutilisation de l'existant est

fortement recommandée Ainsi, deux SI différents utilisant un même type de personnalisation ne nécessitera qu'un seul processus au niveau de la phase de personnalisation (SP) alors que chaque SI nécessiterait une analyse, une conception et une implémentation spécifique.

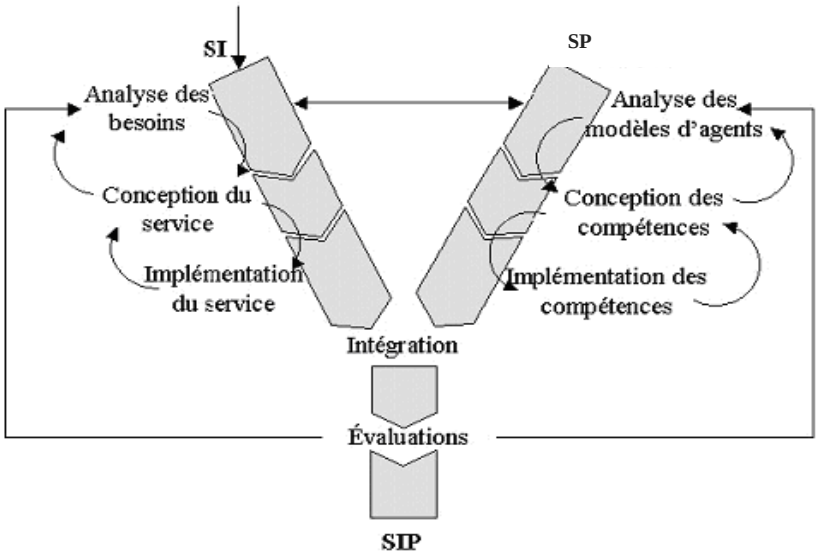


Figure 1.12. Méthodologie de développement d'un SIP.

Le processus de développement d'un SIP passe par les étapes suivantes :

Analyse du service: C'est le point d'entrée de la méthode. Il s'agit d'effectuer une analyse fonctionnelle pour prendre en compte les besoins spécifiques des utilisateurs en terme de services personnalisés. C'est dans cette étape qu'on spécifie les données qui seront échangées entre le SI et le SP.

Conception du service : Il s'agit ici de la conception du service à personnaliser (application externe). L'architecture, les choix techniques, la conception de l'interface utilisateur, etc. sont effectués à ce niveau. Dans le cadre d'une personnalisation d'un service déjà existant, il s'agira essentiellement de la conception de l'interface utilisateur. Ceci suppose que le service existant a été conçu suivant une démarche séparant les données à présenter de l'interface utilisateur.

Implémentation du service: C'est l'étape de la réalisation effective du service. Le service est développé conformément aux modèles conceptuels définis lors de la phase de conception du service. Des tests du service peuvent s'effectuer par simulation des données à fournir au service.

Analyse des modèles d'agents : Il s'agit ici de la spécification des différents modèles d'agents utiles pour les besoins de personnalisation exprimés dans l'étape de l'analyse du service. C'est essentiellement une description des services que doivent fournir les différents modèles d'agents.

Conception des compétences : C'est l'étape de conception des compétences nécessaires pour chaque modèle d'agent décrit dans l'étape précédente.

Implémentation des compétences : Il s'agit de la réalisation effective des compétences. Pour chaque compétence des tests unitaires sont effectués. La création et le déploiement des modèles d'agents, et l'ajout de leurs compétences s'effectuent grâce à l'outil de personnalisation.

Intégration : C'est l'étape d'intégration de SI et SP pour former le SIP. Pour la communication entre les deux applications on peut utiliser une communication par service web qui utilise le protocole de communication SOAP (Simple Object Access Protocol)

Evaluations : Après la réalisation du SIP, des évaluations sont effectuées. Ces évaluations peuvent être de nature qualitative (la qualité de la personnalisation réalisée), quantitative (la performance globale du SIP, les montées en charge), ergonomique ou autres. Ces évaluations peuvent emmener à des révisions au niveau de l'analyse du service et/ou au niveau de l'analyse des modèles d'agents.

1.3.2. Limites de la méthode

- Cette approche s'inscrit dans le cadre de l'accès à une source de données unique : en fait, aucun traitement n'est fait sur la requête de l'utilisateur pour surmonter le problème d'hétérogénéité des sources de données ; la requête de l'utilisateur (SI) est transmise vers le SP – telle quelle est – ; le SP interroge les sources de données en envoyant la même requête ; le filtrage de la réponse en se basant sur les profils des utilisateurs donne naissance à une réponse pertinente qui répond aux préférences et contraintes de l'utilisateur.

→ Sauf les sources de données capables d'interpréter la requête de l'utilisateur seront invoquées.

→ Ne permet pas la personnalisation des systèmes multi-sources **hétérogènes**.

- C'est une approche statique réalisée pour un ensemble de requêtes et non pour chaque requête. Par ailleurs, elle ne vise pas la sélection des sources selon des critères de qualité.

→ Ne permet pas la personnalisation des systèmes à sources de données **distribuées.**

Par ailleurs, cette approche ne prend en compte la personnalisation dans sa globalité, tenant compte à la fois des profils des utilisateurs (centre d'intérêt, préférences, contexte d'exécution de la requête) et des profils des sources de données (méta données décrivant leurs contenus et leurs facteurs de qualité). En effet, les bénéfices de l'accès personnalisé à l'information sont plus visibles dans un contexte distribué où la multiplicité des sources de données conduit à des résultats volumineux, souvent non pertinents et redondants.

1.4. But de notre étude

Elaboration d'une méthodologie de conception des systèmes d'informations personnalisés (SIP) qui possède les avantages de la méthodologie « PerMet » et qui surmonte ses inconvénients, cette méthode doit permettre la personnalisation des systèmes d'information multi-sources hétérogènes et distrubuées.

Elaboration d'une approche de reformulation de la requête utilisateur qui possède les avantages des deux scénarios (E-R et R-E) et qui gomme leurs inconvénients : Cette

reformulation constitue le premier composant d'un système d'information multi-sources à accès personnalisé.

Conclusion

Dans la première section de ce chapitre, nous avons mis en évidence les principales caractéristiques requises par un système d'information personnalisée : intégrer les buts, préférences et capacités des utilisateurs dans le choix des informations à fournir. Ceci présente cependant quelques problèmes, notamment,le problème des données qui sont hétérogènes et physiquement distribuées sur le réseau Internet. Une solution serait que le système soit capable de se déplacer d'une source de données à l'autre, qu'il soit mobile.

Dans la deuxième section de ce chapitre, nous avons présenté trois approches de personnalisation des systèmes d'informations.

Les deux premières concernent la reformulation de la requête de l'utilisateur dans un système multi-sources et la troisième permet la personnalisation des systèmes à données homogènes. En outre, nous avons discuté leurs avantages et inconvénients.

Un première perspective de travail soulevée par cette étude est l'élaboration d'une approche de personnalisation de l'information qui possède les avantages des deux approches: reformulation de la requête et séparation du SI et SP.

Chapitre 2
APPROCHE DE PERSONNALISATION PROPOSEE ET THEMATIQUE DE TRAVAIL

Sommaire

Introduction

Pour profiter des approches de personnalisation de l'information existantes et gommer leurs inconvénients, nous proposons dans ce chapitre un méthodologie de conception de système d'information personnalisés à source de données hétérogènes et distribuées qui résulte de la combinaison des deux approches : celle de la réécriture des requêtes et celle de la séparation entre SI et SP. Dans la deuxième partie de se chapitre, nous présentons la thématique de travail, ainsi que le système de « Portail étudiant » qui représente la partie pratique de ce mémoire.

2.1. Approche proposée

2.1.1. Description générale

Dans le premier chapitre, on a montré que les méthodologies de personnalisation de l'information ne prennent pas en compte la personnalisation de l'information dans sa globalité, tenant compte à la fois des profils des utilisateurs (centre d'intérêts, centre d'exécution de la requête, contraintes temporelles spatiales,…) et des profils des sources de données (métadonnées décrivant leurs contenus et leurs facteurs de qualité). Pour prendre en compte se compromis entre les deux profils nous proposons une méthodologie de conception qui se base sur la séparation entre le SI et le SP pour la personnalisation du point de vue profil de l'utilisateur et la notion de médiation de l'information pour surmonter le problème d'hétérogénéité et de distributivité des sources de données.

Pour mieux distinguer les parties du SI qui sont à personnaliser, nous considérons un SI comme un ensemble de services. Un service personnalisé correspond alors à une unité fonctionnelle permettant une interaction homme-machine personnalisée [Anli et al., 05].

Pour répondre à l'objectif de distributivité et d'évolutivité, nous préconisons, comme [Deschaine et al., 00] ou [Dickinson et al., 03] ou [Anli et al., 05], l'utilisation de SP à base d'agents logiciels. En effet, la distributivité et l'évolutivité pourraient être facilitées grâce aux caractéristiques intrinsèques des agents logiciels. La distributivité est assurée grâce aux caractéristiques d'autonomie, de communicabilité et de mobilité des agents logiciels. L'évolutivité est favorisée grâce aux caractéristiques d'adaptabilité et de reproductibilité.

Pour le développement d'un SIP multi-sources, on suit un modèle de développement suivant cinq parties (voir Figure 2.1.). La partie SI concerne le développement d'un service de SI. La partie SP concerne l'adaptation et la configuration d'un SP à base d'agents logiciels pour répondre aux objectifs de personnalisation du service et la partie MEDIATEUR correspond au développement d'un médiateur de sources hétérogènes.

Les trois parties SI, SP et MEDIATEUR suivent un modèle de développement classique pouvant se dérouler en parallèle. Par la suite SP et MEDIATEUR se rejoignent pour former un système de personnalisation multi-sources. L'intégration de ce système résultant et SI donne naissance à un système d'information personnalisé et multi-sources Le modèle proposé est itératif et incrémental. Le modèle de développement doit être parcouru plusieurs fois pour aboutir à un service personnalisé opérationnel.. Cependant, il n'est pas nécessaire de spécifier tous les services

à personnaliser dans un SI. Les autres services peuvent être spécifiés et développés au fur et à mesure des besoins.

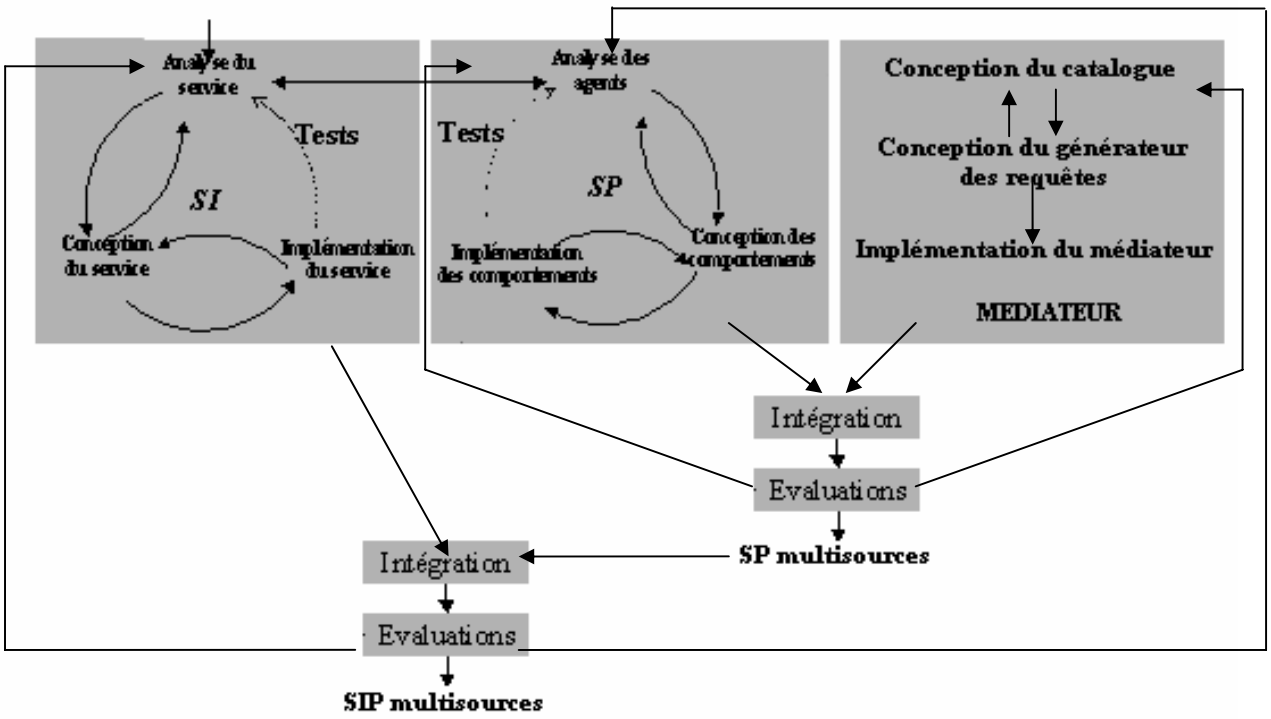


Figure 2.1 Le modèle de développement d'un SIP multi-sources.

2.1.2. Les différentes phases de l'approche proposée

Pour toutes les phases de développement du SI et SP en se base sur la méthodologie PerMET [Anli 06]. Nous rappelons par la suite les principes de ces différentes phases.

2.1.2.1. Analyse du service

Cette phase présuppose qu'une étude préliminaire a été effectuée et que le service personnalisé à mettre en œuvre est identifié. Cette phase suit le modèle d'analyse de 2TUP pour répondre aux contraintes d'évolution du service. L'analyse du service est donc effectuée suivant un aspect fonctionnel et suivant un aspect technique pouvant se dérouler en parallèle. Evidemment, les activités à réaliser pour chaque aspect dépendent fortement du projet.

2.1.2.2. Conception du service

La conception vise à préciser le modèle d'analyse de telle sorte qu'il puisse être implémenté avec les éléments de l'architecture. Il s'agit d'exprimer les modèles statiques et dynamiques qui seront traduits directement sous forme de codes exécutables dans la phase d'implémentation du service.

2.1.2.3. Implémentation du service

C'est l'étape de la réalisation effective du service. Le service est développé conformément aux modèles conceptuels définis lors de la phase de conception du service. Pendant l'implémentation, le développeur veillera à ce que le modèle de conception reflète exactement le code informatique produit. Des tests du service peuvent s'effectuer par simulation des données (conformément aux modèles de données définis dans la phase d'analyse) à fournir au service (ces données seront issues du SP lorsque la phase d'intégration sera achevée). Ces tests peuvent aboutir à une révision du modèle d'analyse du service ce qui nécessitera une autre réalisation des différentes phases de la partie SI et aussi de la partie SP.

2.1.2.4. Analyse des agents

Il s'agit ici de l'analyse des différents agents utiles pour les besoins de personnalisation. Cette phase peut démarrer à l'issu de l'étape d'analyse fonctionnelle de la phase d'analyse du service. Les modèles agents, les comportements, les acquaintances et les déploiements des agents sont identifiés à ce niveau. Les activités à réaliser dans cette phase sont : (voir Figure 2.2.)

Analyse des modèles d'agents : cette étape consiste à identifier les types d'agents nécessaires pour la réalisation des besoins fonctionnels définis dans la phase d'analyse du service. Les modèles d'agents peuvent être identifiés en suivant les règles ci-dessous :

1. un modèle d'agent par fonctionnalité attendue du service.
2. un modèle d'agent par type plate-forme d'interaction.
3. un modèle d'agent par ressource externe dont le système doit accéder (intéragir).

Analyse des comportements : il s'agit, ici, d'analyser pour chaque agent les comportements adéquats pour la réalisation de rôle. Cette analyse est assez délicate car il n'y a pas de méthode objective pour déterminer la granularité nécessaire à la décomposition des comportements. Cela dépendra donc de l'expérience et du savoir-faire du développeur. Une règle principale à appliquer consiste à décomposer un comportement tant qu'une tâche associée à ce comportement peut être associée à un autre comportement. Par exemple, prenons le comportement *« enregistrer le choix de l'utilisateur»*. Ce comportement nécessite l'exécution de la tâche « charger le profil de l'utilisateur> puis l'exécution de la tâche « mettre à jour le profil ». En analysant d'autres comportements comme par exemple *« envoyer un mail »*, la tâche « charger le profil de l'utilisateur » (pour récupérer l'adresse électronique de l'utilisateur) est également nécessaire. Il revient donc que « charger le profil de l'utilisateur » soit considéré comme un comportement.

Identification des acquaintances : cette étape décrit les relations entre les différents agents et leurs croyances sur les capacités des uns et des autres. Les relations d'interactions de chaque agent avec les autres y sont modélisées.

Information de déploiement : il s'agit d'une description des différentes localisations physiques où les agents logiciels vont s'exécuter. Cette description peut aussi bien concerner les informations de localisation statique (l'agent est localisé à un même endroit et ne change jamais de place) que les informations de localisation dynamique (l'agent peut changer d'endroit dynamiquement suivant les tâches qu'il veut accomplir).

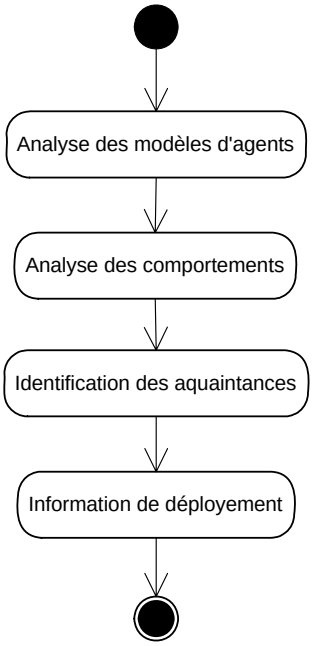


Figure 2.2. Les étapes de l'analyse des agents.

2.1.2.5. Conception des comportements des agents

C'est la phase de conception des comportements des agents identifiés dans la phase ci-dessus. Il s'agit d'affiner les modèles issus de la phase d'analyse des agents pour aboutir à des modèles qui puissent être directement traduits en code.

2.1.2.6. Implémentation des comportements des agents

Il s'agit de la réalisation effective des comportements des agents. Pour chaque comportement des tests unitaires sont effectués. La création des agents, leur déploiement et l'intégration de leur comportement se feront grâce à des outils d'administration des agents. Ces outils sont généralement fournis par les plates-formes de développement d'agents. Par exemple, MADKit et Jade fournissent des outils graphiques permettant de créer, de déployer et d'interconnecter (relations d'acquaintance ou de simples relations de communications) les agents logiciels. Les outils actuels ne permettent pas de changer dynamiquement les comportements des agents en cours d'exécution pour répondre à notre objectif d'évolutivité du SP mais les caractéristiques des agents prédisposent ces outils à intégrer cette fonctionnalité. D'ailleurs, l'outil fournit par Jade prévoit cette fonctionnalité (la fonctionnalité n'est pas encore

opérationnelle mais les interfaces graphiques sont déjà développées). Cette fonctionnalité devrait donc être disponible dans l'outil de Jade très bientôt.

Après avoir déployer les agents, des tests du SP doivent être effectués pour vérifier si le système multi-agents formant le SP répond bien aux objectifs définis dans la phase d'analyse des agents. Ces tests s'effectueront par simulation des données envoyées par le service (au niveau du SI) au SP. Ces tests peuvent aboutir à une révision du modèle d'analyse des agents ce qui nécessitera une autre réalisation des différentes phases de la partie SP.

2.1.2.7. Conception du catalogue

La conception du catalogue comprend quatre étapes :

- Concevoir le fichier d'information sur les sources locales
- Concevoir l'ontologie su schéma global
- Concevoir le mapping entre ontologies globale et locales sous forme de table de correspondance
- Faire correspondre les synonymes et les hyperonymes entre les concepts des schémas globaux et ceux des schémas locaux.

2.1.2.8. Conception du générateur de requêtes/sous-requêtes

En fait, le mapping constitue un traitement clé dans le processus général. Il sera utilisé pour réécrire la requête initialement exprimé en fonction du schéma global, en des sous-requêtes exprimées chacune, en fonction du schéma local de la source qui l'exécutera.

Ce module prend en entrée la requête représentée sous forme XML. Deux phases sont nécessaires à cette étape :

- Prétraitement et génération des sous-requêtes
- Génération des sous-requêtes comportant juste la portion d'information nécessaire.

2.1.2.9. Implémentation du médiateur

Il s'agit de la réalisation effective des composants du médiateur. Le médiateur est développé conformément au modèle conceptuel défini lors des phases de conception du catalogue et du générateur de requêtes. Des tests doivent être effectués pour vérifier si le système de médiation permet de surmonter les problèmes d'hétérogénéité des sources.

2.2. Thématique : Espace de Services PERsonnalisés pour l'Etudiant (ESPERE)

2.2.1. Quelques exemples de portails étudiants

- **Portail étudiant de l'éduction nationale**[1] : moteur de recherche de formations, guide des études et de la vie étudiante en France. Ce portail offre le site Hand-U qui cherche à offre l'information pertinente pour la vie d'un étudiant handicapé : nom des responsables d'accueil,

[1] www.etudiant.gouv.fr

aides diverses auxquels il peut prétendre, le lien avec le CNOUS et le CROUS pour la restauration et l'hébergement, transport, textes officiels, adresses utiles,…

→ Offre une personnalisation pour une catégorie prédéfinie et spécifique des utilisateurs du système.

- **Portail étudiante de l'université de Lyon**[2]
  Il offre pas de mal de fonctionnalité qui se voient utiles pour l étudiant, telles que :
  - L'inscription à distance
  - Les annonces de logements
  - Des annonces diverses
  - Les nouvelles culturelles : cinéma, festivals, soirées, rencontres
  - Recherche documentaire

  Cependant toutes ces fonctionnalités ne prend pas en charge les contraintes et préférences de l'étudiant (elles sont offertes d'une manière statiques et uniforme pour toutes les utilisateurs du système)
- **Le site portail 100% étudiant 100% jeune diplômé- universités**[3] : portail étudiant proposant es informations générales sur l'orientation, l'emploi, les stages, le logement et les associations étudiantes.
- **Portail des étudiants ingénieurs de l'UTC**[4] **:**
  - Propose des généralités sur la vie de l'étudiant, la vie associative et des insertions professionnelles
  - Offre les actualités et nouveautés
  - Donne des liens utiles qui peuvent intéressés l'ingénieur.
- **Portail d'orientation**[5] **:** c'est un portail pour résoudre toutes les questions d'orientation et de formation. Il offre une recherche libre, ou personnalisée de formation et de métier
  - → Personnalisation de la recherche
  - → Pas d'information temps réel.

Les exemples de portails étudiant sont énormes –nous avons cités quelques unes- cependant peu entre eux qui offrent quelques fonctionnalités personnalisées qui répond aux contraintes de son utilisateur et à nous connaissances, actuellement aucun portail étudiant n'est totalement personnalisé, en d'autre terme, offre l'information temps réel à son utilisateur si et seulement si, elle peut l'intéresser et répondre à ses préférences et contraintes.

La partie pratique de cette étude, s'inscrit dans le cadre de la conception et éventuellement la réalisation d'un portail personnalisé pour l'étudiant « ESPERE» .

2.2.2. Contexte du projet.

2.2.2.1. Les objectifs généraux du projet

A travers un espace de services portail, on entrevoit la réalité d'une aide à l'étudiant par la pertinence de l'information délivrée. L'étudiant ne souhaite en effet avoir à disposition que

[2] www.etud.univ-lyon2.fr
[3] www.capcamus.com
[4] www.utc.fr
[5] www.orientation-formation.fr

d'informations qui l'intéressent directement. Un « système d'information », comme son nom l'indique, est destiné à fournir de l'information à un utilisateur. En fait, il devrait, idéalement, permettre à l'utilisateur de récupérer de l'information à partir de données auxquelles a accès le système. Or, cette transformation des données en information, à savoir cette plus-value apportée aux données qui sont triées, classées, validées et personnalisées, est bien souvent négligeable : le système laisse à l'utilisateur la charge de retrouver l'information qui l'intéresse dans la masse de données qui lui est fournie.

Nous essayons de remédier à cela dans le contexte de l'information aux usagers. En particulier, notre objectif est, d'une part, d'aider l'utilisateur dans sa démarche de recherche d'information et , d'autre part, de lui fournir un résultat personnalisé, et no une large diffusion de toutes les informations disponibles. Personnaliser l'information consiste à présenter toue l'information nécessaire et uniquement l'information nécessaire en fonction de son destinataire.

2.2.2.2. Stratégies et méthodes pour y répondre

Parmi les pistes de travail proposées dans le domaine de l'information multimodale, nous nous intéressons ici particulièrement à « l'articulation entre l'individuel et le collectif », à savoir l'adaptation de l'information fournie aux besoins spécifiques de l'usager.

Nous souhaitons mettre en œuvre et évaluer des services liés à la personnalisation des informations. Ces services seraient centrés sur l'usage des services et la mobilité au quotidien, dans un but d'accompagnement du jeune étudiant dans son « univers » et ses activités.

Plus précisément, notre objectif est de développer un démonstrateur, désigné par « ESPERE », capable de :

- Extraire des gisements de données de l'information pertinente pour les étudiants
- Organiser la mobilité quotidienne en fonction de l'agenda de l'étudiant fin de gagner du temps et mieux organiser ses activités et son déplacement,
- Présenter l'information de manière immédiate et naturelle. Seront distinguées l'information statique et l'information temps réel,
- Diffuser l'information sur plusieurs supports, tout en assurant une présentation ergonomique et un contenu homogène.

Nous s'appuyons principalement les techniques des systèmes multi agents(SMA). Il s'agit d'agents logiciels intelligente, mais aussi avec des capacités d'échanges d'information et de complémentarité.

La philosophie des systèmes multi agents fait appel au principe d la délégation. Ainsi, au lieu d'avoir un seul système, très intelligent et autonome qui prend en charge la résolution de l'intégralité d'un problème par la perception de son environnement, on considère plusieurs agents, et à chaque agent, on délègue la résolution d'une partie du problème. Ainsi la solution est obtenue grâce aux comportements individuels et aux interactions entre les agents. Les systèmes multi agents représentent alors une nouvelle approche pour l'analyse, la conception et l'implantation des systèmes informatiques complexes.

Au travers d'un mode de coopération prédéfini, ces agents communiquent dans le cadre d'une organisation pour réaliser des objectifs. Par exemple, un agent pourra chercher de l'information en transports collectifs, un autre agent de l'information routière, et un troisième ne présentera que l'information pertinente en fonction du profil du demandeur en faisant appel à une coopération prédéfinie entre les deux premiers.

2.2.2.3. Domaine d'Application

L'objectif visé est le développement d'un démonstrateur de « portail Etudiant » permettant de valider les modèles et les architectures étudiés. Ce portail étudiant constituera un système d'informations personnalisées qui intégrera dans un premier temps deux types de services :

- l'accès aux informations de base « universitaires » et des ressources pédagogiques numériques
- l'accès à des services connexes mais aussi sur la mobilité, dans un but d'accompagnement de la population estudiantine dans leurs « univers » et leurs activités

2.2.2.4. Définition et création des services

La première phase consistera à analyser et définir précisément les services visés et développer une première version des applications permettant de mettre en œuvre ces services.

- **Analyse et définition fonctionnelle des services d'accompagnement**

L'innovation principale du système d'information multimodale et personnalisée consiste à présenter certaines informations à bon escient, au bon moment, de façon ciblée pour l'utilisateur. Nous tritons ici l'univers de l'étudiant et par ce biais observons le monde des études, des loisirs, de la famille, des achats, etc.

L'accompagnement personnalisé d'un étudiant dans ses activités quotidiennes suppose la connaissance d'informations personnelles quant à son profil afin de lui rendre les meilleurs services « sur mesure ». Les informations utiles correspondent aux contextes dans lesquels se situe l'utilisateur :

- **Contextes spatiaux :** typiquement, les lieux fréquentés (domicile études, domicile parents, lieu d'études, lieu de stages,…), comprenant également les lieux/moyens d'accès à l'information universitaire, les canaux de communications disponibles (mobile, téléphone fixe, internet,etc)
- **Contextes temporels :** l'agenda, au sens large, c'est-à-dire incluant à la fois les horaires liés aux études ou stages, et les horaires des rendez-vous, des activités sportives, culturelles, etc
- **Contextes identitaires :** qui est-il/ elle ? il s'agit ici de cerner le profil de l'utilisateur sur la base de quelques données personnelles (âge, adresses, handicap, la possession de cartes de vie quotidienne étudiant, abonnement de transports collectifs, club sport, fidélisation magasins, bancaire, etc)

Ces informations doivent permettre d'offrir un service pertinent en fonction de l'univers quotidien de l'étudiant. La définition fonctionnelle vise donc à définir les contextes précis qui seront pris en compte.

- **Ecriture et validation logicielles**

Le système disposera de différents moyens pour accéder aux informations nécessaires :

- Certaines informations peuvent être demandées directement à l'utilisateur, lors de son inscription au service ou lorsque le système en a besoin
- Certaines informations peuvent être connues indirectement, en lieu avec d'autres bases de données. Par exemple, les données relatives à un événement de modification d'un emploi du temps, à un événement culturel , à une nouvelle ressource pédagogique, etc., seront connus via des sites Internet et ce, de manière transparente pour l'utilisateur
- Enfin certaines informations peuvent être déduites par le système sur la base des connexions passées. Par exemple, le système peut apprendre des préférences ou habitudes relatives à des contextes précis en termes d'activités. On parle alors d'apprentissage automatique.

Quant à la réalisation des services visés, nous nous baserons sur des travaux théoriques antérieurs proposant une architecture à base d'agents logiciels. Les agents logiciels offrent des possibilités d'assistance aux utilisateurs (agent d'interface), d'accès à des bases de données hétérogènes et distribuées (agent mobile) et d'apprentissage automatique (agent apprenant), dans le cadre d'une coordination en vue de réaliser une tâche commune.

- **Agent de recherche d'information :**

Ces agents ont pour objectif général la recherche d'information dans le web. Ils peuvent utiliser des méthodes de recherche d'information issues de la découverte automatique de données, le datamining, etc

- **Agent assistance de l'utilisateur :**

Cette partie est assurée par un agent installé sur le poste utilisateur pour permettre une interaction directe entre l'utilisateur et le SP. Par exemple, l'utilisateur final pourra spécialiser explicitement au SP si un document qu'il est entrain de visualiser le plait ou non. Cela permettre au SP de connaître mieux les préférences de l'utilisateur. Cet assistant permettra aussi à l'utilisateur de resté connecté au SP pour une observation automatique plus personnalisée de ses interactions pour une assistance plus accrue.

- **Agent de gestion de profil :**

Ces agents assurent la gestion des profils utilisateur. La centralisation de la gestion des profils utilisateur par des agents va permettre une meilleure ré-utilisation du profil. Les informations communes utilisées par les agents assistants et les agents de recherche d'information. Par exemple, sont stockées une seul fois. Si un utilisateur aime le cinéma, par exemple, cette préférence sera gérée au niveau de la gestion de profil. Si un agent assistant ou un agent de recherche d'information a besoin de cette donnée, il transmettra sa requête à l'agent coordination qui lui transmettra la réponse de l'agent gestion de profil. Les méthodes d'apprentissage communes aux différents agents et la mise à jour des profils sont aussi assurées à ce niveau.

Supposons que l'agent coordination reçoive une requête : »quels sont les trois films préférés d'un utilisateur A ? ». il transmettra sa requête à un agent de gestion de profil qui se chargera de lui répondre. La réponse peut être basée sur des modèles de raisonnement plus ou moins élaborés

- **Agent de coordination :**

Toutes les interactions passent par l'intermédiaire de cet agent. Il fournit une interface homme-machine permettant l'administration et la gestion du SP. Par exemple, la création et l'acquisition des compétences d'un agent de gestion de profil s'effectuent par l'intermédiaire de cet agent de coordination. La communication inter-agent s'effectue par l'intermédiaire de l'agent de coordination.

Par exemple, si un agent assistant recherche les préférences par rapport à un profil utilisateur donné, sa requête est transmise à un agent de gestion de profil par l'intermédiaire de l'agent coordination. L'agent coordination fournit certaines de ces compétences au moyen de services web. Les applications externes au système de personnalisation communiquent avec le système de personnalisation au travers le SOAP¨.

L'intérêt de ces travaux pour les systèmes d'information multimodale est qu'ils proposent des modèles réutilisables en conception de systèmes de ce type. Par ailleurs, en restant au niveau conceptuel de la modélisation, ils ne présupposent pas de choix technologique spécifique sous-jacent. Ainsi nous nous engagerons à mettre en œuvre un démonstrateur opérationnel totalement implanté et maintenu à partir de logiciels libres.

La mise en œuvre de la personnalisation suppose de la personnalisation suppose la représentation de connaissances relatives aux utilisateurs du système.

L'univers de l'utilisateur sera décliné selon les trois contextes évoqués précédemment :

- L'univers identitaire comporte une forte composante de données statiques, fournies directement par l'utilisateur à l'inscription au service. L'accent sera mis sur la création d'interfaces conviviales afin d'éviter de rendre ennuyeux ce recueil de données
- L'univers spatial, comporte les lieux fréquentés, sera organiser de façon à faire émerger des repères majeurs, les principaux lieux de vie. Les capacités d'apprentissage seront ici utilisées afin de crées progressivement cette « hiérarchisation » des lieux selon les demandes formulées.
- L'univers temporel est fortement lié à l'agenda de l'étudiant. Dans ce domaine également, il est envisagé d'organiser les connaissances, dans le but de distinguer les activités impliquant des contraintes temporelles forte (par exemple, des heures de cours) des activités moins contraintes et qui peuvent de ce faire être « décalées » dans le temps (par exemple, l'heure d'accès aux commerces).

Le démonstrateur sera réalisé de façon incrémentale afin d'intégrer progressivement des fonctionnalités de niveaux de plus en plus élevés :

- Le démonstrateur doit porter des services de base liés à l'information transport, tels que la planification d'itinéraires personnalisés à la demande de l'utilisateur.

- Ensuite il s'agit d'intégrer la prise en compte de l'agenda dans les services proposés. Tels que l'optimisation des déplacement prévues.
- Enfin, le démonstrateur dont permettre d'effectuer des propositions de manières dynamiques, tels que l'offre de re-planification liée à des modifications de l'agenda, la prise en compte d'événements ponctuels ou une nouvelle proposition relative à une « opportunité » de déplacement pour un motif en rapport étroit avec le profil de l'utilisateur.

Nous proposons des solutions logicielles permettant de supporter cette personnalisation, de même qu'un ensemble d'interfaces homme-machine dédiées à celle-ci. Les fonctionnalités proposées pourront être implantées grâces aux agents logiciels. Les interfaces développées devront être conviviales, intuitives et innovantes.

En parallèle au développement de l'application, des scénarios-types seront définis, dans le but d'illustrer les services proposés et de procéder aux évaluations.

2.3. Description générale de notre système

Nous proposons pour la réalisation d'un portail étudiant personnalisé, un système composé de 3 parties indépendantes, inter-opérables et communicantes (voir Figure 2.3.)

- Partie I : Le système de personnalisation « ESPERE »
- Partie II : Le médiateur « MedESPERE »
- Partie III : Le système d'informations « ESPERE »

Ces trois parties sont décrites dans la suite de ce rapport.

Notre portail est un système évolutif et incrémental. Il permet d'ajouter les fonctionnalités sous forme de services. Pour notre étude, nous nous limitons aux quatre services suivants : la recherche documentaire, la recherche d'itinéraire, la recherche d'emplois et rattrapages et la recherche de stages.

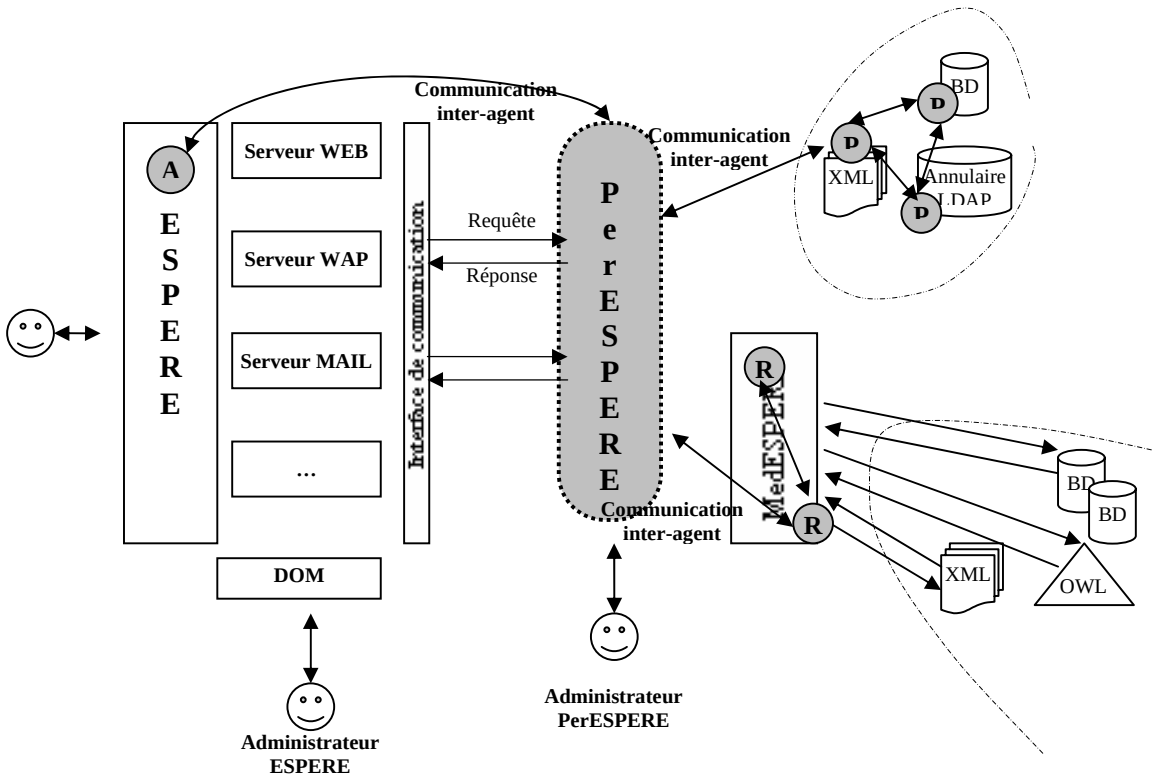


Figure 2.3. Architecture générale du système « Portail étudiant ».

Le SP et le médiateur sont localisés sur le même serveur, tandis que la communication entre le SI et le SP est assurée grâce au protocole de communication SOAP intégré dans le service web assurant l'échange et l'exécution des requêtes entre les deux parties SI et SP.

Conclusion

Ce chapitre décrit notre contribution pour la personnalisation de système d'information à sources de données hétérogènes. Nous avons proposé une méthodologie qui compose tel système en trois parties indépendantes et inter-communicante : le système d'information, le système de personnalisation et le système de médiation.

Notre approche propose un modèle de développement itératif, incrémental et permet une réalisation parallèle des phases spécifiques liées au développement des services, des phases spécifiques liées à la personnalisation ainsi que les phases spécifique liées au système de médiation.

Deuxième partie

Chapitre 3
PerEspere, un système de personnalisation sur mesure pour le projet ESPERE

Sommaire

Introduction

Pour faciliter la réalisation d'un système de personnalisation de l'information dédiée pour l'étudiant, nous proposons PerEspere, un système de personnalisation sur mesure pouvant s'utiliser pour le développement d'un espace de Services PERsonnalisés pour l'Etudiant (ESPERE).

Après une présentation générale de la plate-forme utilisée pour le développement des agents logiciels composant PerEspere, ce chapitre décrit l'architecture générale et la conception de notre système de personnalisation ; les différentes fonctionnalités permettant l'évolutivité et la distributivité de PerEspere sont exposées.

3.1. Présentation globale de la plate-forme d'agents « Magique »

De nombreuses plate-formes existent pour le développement d'applications à base d'agents logiciels. Le choix d'une plate-forme est généralement motivé par sa facilité de prise en main et par sa facilité de mise en œuvre de différents modèles d'agent. En effet, certaines caractéristiques relatives aux agents sont plus ou moins faciles à mettre en oeuvre suivant la plate-forme de développement utilisée. Par exemple, il est beaucoup plus aisé de développer des agents mobiles avec la plate-forme VOYAGER de Recursion Software6 qu'avec la plate-forme JACK d'Agent Oriented Software7 [Howden et al., 01] qui, elle, est bien adaptée pour le développement d'agents de type BDI (Belief-Desire-Intention).

Pour développer notre système de personnalisation, nous avons opté pour la plate-forme Magique (Multi-AGent hiérarchIQUE) [Routier et al., 01] de l'équipe Système Multi-Agents et Comportements (SMAC) du Laboratoire d'Informatique Fondamentale de Lille (LIFL)8. Ce choix a été établi parce que certains principes de Magique, que nous détaillerons par la suite, nous ont apparu pertinents pouvant faciliter la conception de notre système de personnalisation.

3.1.1. Définitions et concepts

La plate-forme Magique se présente sous forme d'une API[9] Java qui facilite la conception de système multi-agents. Magique se base sur un ensemble de concepts pour la construction d'un système multi-agents. Les définitions de ces concepts qui sont présentées dans cette partie sont issues de [Routier et al., 01] et bien évidemment, de la rubrique consacrée à Magique de la page web de l'équipe SMAC. Magique se repose essentiellement sur le concept de *Compétence*.

Définition 1 : *une **compétence** désigne un ensemble cohérent de capacité.*

Une compétence précise donc les tâches que l'agent peut effectuer. Elle décrit une série d'activités que l'agent doit réaliser pour effectuer une opération précise. La compétence définit donc le comportement d'un agent. Les compétences peuvent être développées indépendamment de tout agent. Elles sont ensuite ajoutées aux agents pour définir leurs comportements. Par conséquent, les capacités de communications, de reproduction, de mobilité, d'apprentissage, etc. peuvent être associées à des compétences que l'agent peut en disposer ou non.

[6] http://www.recursionsw.com/
[7] http://www.agent-software.com
[8] http ://www.lifl.fr/SMAC
[9] Application Program Interface

Au niveau de l'implémentation, une compétence correspond à une classe Java héritant de la classe *MagiqueDefaultSkill* fournie par l'API Magique dont les méthodes publiques correspondent aux tâches que l'agent peut effectuer et peut mettre au service des autres agents composant le système multi-agents.

Définition 2 : *un **agent** est une entité douée de compétences.*

Un agent est donc une entité possédant un certain nombre de compétences. Ces compétences permettent aux agents de jouer un rôle au sein du SMA. Les compétences d'un agent peuvent évoluer dynamiquement (par apprentissage/acquisition) au cours de son existence.

Pour préciser cette définition les auteurs de Magique propose la notion *d'agent atomique*.

Définition 3 : *un **agent atomique** est une entité douée de deux compétences : une pour interagir et une pour apprendre de nouvelles compétences. Un agent est un agent atomique qui a appris des compétences au travers de communications.*

Les auteurs de magique soulignent la nécessité d'une capacité d'interaction et d'apprentissage (de compétences) pour un agent logiciel. *« Sans la "compétence d'acquisition", un tel agent ne serait qu'une coquille vide incapable de faire quoique ce soit. Sans la "compétence de communication", un agent est isolé du "reste du monde" et perd de ce fait tout intérêt ».*

Les agents Magique possèdent donc les deux compétences de communication *ConnectionSkill* et d'acquisition de compétence *AddSkillSkill.* Ils peuvent communiquer en asynchrone (*perform* / *ask*) et en synchrone (*askNow*).

L'une des particularités de Magique se situe au niveau des principes d'oubli et d'échange de compétences. Un agent peut oublier une compétence s'il juge, par exemple, que cette compétence est obsolète. Il peut aussi enseigner à un autre agent une compétence dont il dispose ou demander auprès d'un agent de lui enseigner une compétence pour pouvoir, par exemple, effectuer une tâche. Ces principes sont implémentés dans Magique sous forme de compétences et les agents Magique disposent de ces compétences, par défaut, à leur création.

Organisation : Magique utilise une structure organisationnelle hiérarchique. Les agents feuilles sont appelés « *spécialistes* » et les autres appelés « *superviseur* ». Cette structure décrit les liens d'accointances et définit le support de communications entre les agents. Magique se base sur les relations d'accointances pour le routage par défaut des messages. La communication est donc essentiellement verticale. Cependant, il est toujours possible d'établir une relation directe d'accointances (donc de communication) entre les agents (voir Figure 4.1).

Magique se base sur la structure hiérarchique pour réaliser un mécanisme de délégation de tâche. Lorsqu'un agent doit exécuter une tâche et qu'il n'a pas la compétence nécessaire, il regarde s'il a une accointance particulière pour cette compétence et lui demande de réaliser la tâche pour lui. Sinon, il la fait exécuter par un membre de sa hiérarchie qui possède cette compétence. Sinon, il demande à son superviseur de lui chercher quelqu'un de compétent. Son superviseur re-applique ce mécanisme récursivement. Lorsque aucun agent du SMA ne dispose de la compétence adéquate, la requête est stockée au niveau de l'agent racine jusqu'à ce qu'un des agents formant le SMA acquière cette compétence pour exécuter la tâche.

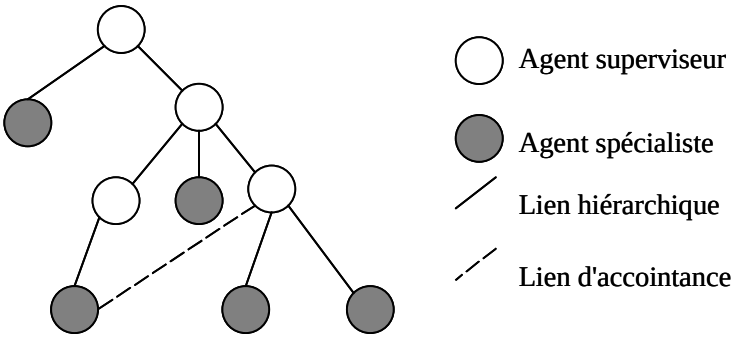


Figure 3.1. Structure organisationnelle hiérarchique.

3.1.2. Conception et Interaction avec les agents

Un environnement graphique permettant la conception et le déploiement des agents Magique est disponible. Cet environnement permet la construction des agents par la définition des compétences que doit disposer chaque agent, l'organisation hiérarchique de ces agents, la précision de la plate-forme d'accueil où ils doivent s'exécuter et leur déploiement. Toutes ces actions sont effectuées avant l'exécution effective des agents.

Cet environnement intègre un interpréteur de commande Java permettant l'interaction avec les agents en exécution (voir Figure 4.2.). C'est une interaction textuelle et l'utilisateur voulant interagir avec les agents doit utiliser des commandes spécifiques. Aussi il n'est pas possible d'utiliser cet outil (tel qu'il est fournit actuellement) pour interagir avec des agents qui ne soient pas lancés au travers de l'environnement graphique de Magique. Et lorsque l'environnement graphique est fermé, il n'est plus possible d'interagir avec les agents créés auparavant même en relançant l'environnement graphique.

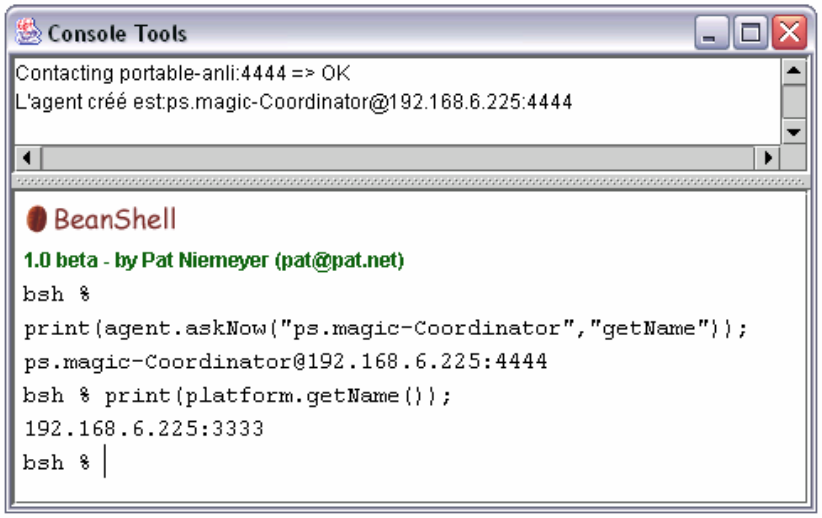


Figure 3.2. Outil d'interaction textuelle avec les agents.

3.2. Architecture générale et conception de PerEspere

La caractéristique essentielle que doit avoir le Système de Personnalisation (SP) sont : la possibilité de communication avec des applications externes (non nécessairement à base d'agents logiciels), essentiellement, les services du SI « ESPERE ». Il est donc naturel que l'architecture de PerEspere comporte un agent permettant cette communication (agent de communication) [Anli .05]. Pour faire le lien entre les différents agents du SP, un autre agent (agent de coordination) a été défini. En effet, puisque les agents sont complètement autonomes et peuvent se localiser un peu partout dans le réseau il est nécessaire de disposer d'un référentiel qui permettra au développeur de localiser les agents et éventuellement d'interagir avec eux (faire évoluer leur compétence, changer leur localisation, etc.). Cet agent de coordination intervient aussi pour la transition des différents messages que les agents peuvent s'échanger pour répondre à un objectif global du SP. L'architecture générale de PerESPERE est donc composée de différents agents. La Figure 4.3. présente l'architecture générale de PerESPERE et ses interactions avec le système ESPERE.

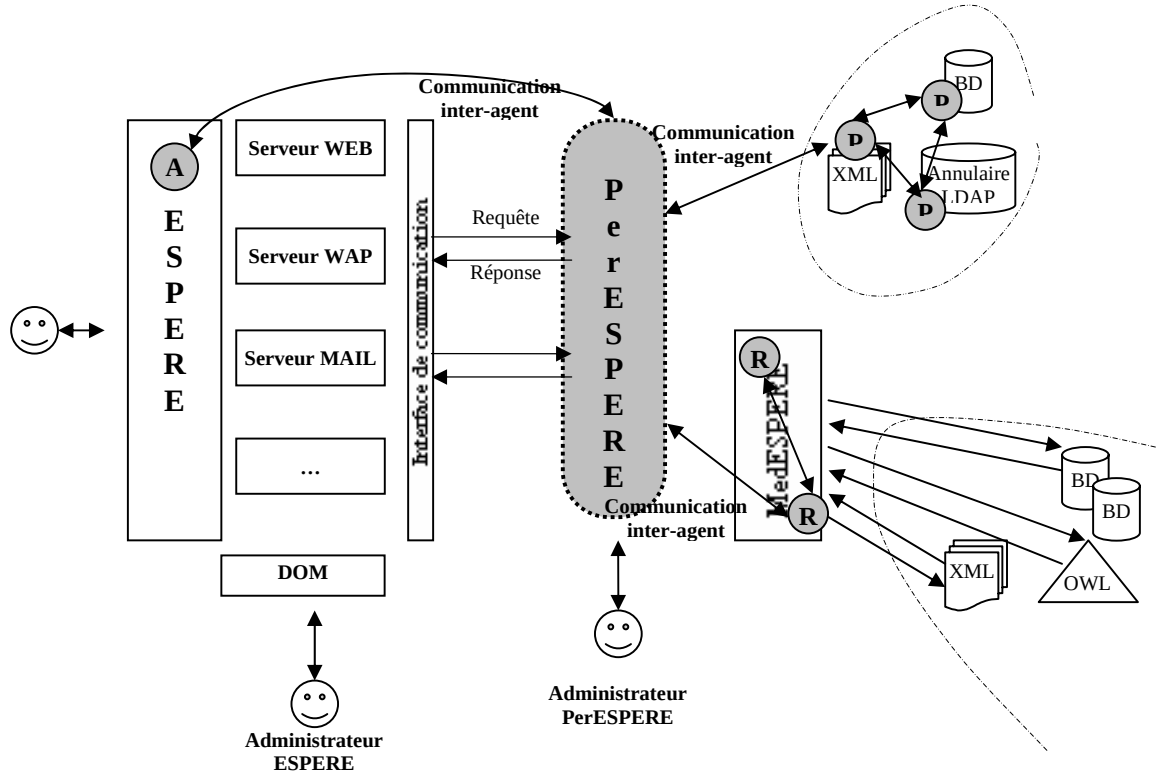


Figure 3.3. Architecture générale de PerEspere.

Les trois agents de *communication*, de *coordination* et *d'administration* (ce sont les agents contenus dans l'ellipse à fond grisé) forment le noyau de PerESPERE. PerESPERE se base sur ces trois agents pour l'intégration de services personnalisés. Les autres agents, que nous appelons *agents applicatifs*, (A comme Assistant, P comme Profil, R comme Recherche) sont des exemples d'agents (ce sont les modèles d'agents les plus utilisés pour la construction de systèmes d'information personnalisés) qui pourraient être définis pour répondre à des objectifs précis suivant un projet particulier. La seule contrainte que doivent respecter ces agents consiste d'avoir un lien d'accointance avec l'agent de coordination pour qu'ils puissent être localisés pour des besoins

d'évolutivité. Il n'est pas nécessaire que cela soit un lien d'accointance direct. Il suffit qu'il existe un « chemin d'accointances » reliant un agent avec l'agent de coordination.

3.2.1. L'agent de coordination

L'agent de coordination permet de faire le lien de communication entre les différents agents composant PerESPERE. L'agent de coordination assure trois rôles : la coordination des messages échangés entre les différents agents applicatifs et la coordination des messages échangés entre les applications externes et les agents applicatifs [Anli .06].

Coordination des messages échangés entre les différents agents applicatifs

L'agent de coordination coordonne les messages échangés entre les différents agents applicatifs. PerESPERE préconise de ne gérer au niveau de cet agent que la coordination des messages échangés entre agents de domaine d'activités différentes. La coordination entre agents du même domaine d'activités est déléguée à un autre agent applicatif. Par exemple, l'échange de message entre l'agent responsable des recherches est assuré par le coordinateur, par contre ceux échangés avec l'agent de recherche de module sont la responsabilité de l'agent de recherche lui-même. Cela évite de surcharger l'agent de coordination et permet d'avoir une meilleure structuration du système. L'analyse des modèles d'agents permet de distinguer les différents domaines d'activités des agents. Un modèle d'agent est une abstraction d'un type d'agent effectuant les mêmes activités. Les agents issus d'un même modèle d'agent disposent donc des mêmes compétences. Ces modèles d'agent peuvent être organisés suivant une structure hiérarchique.

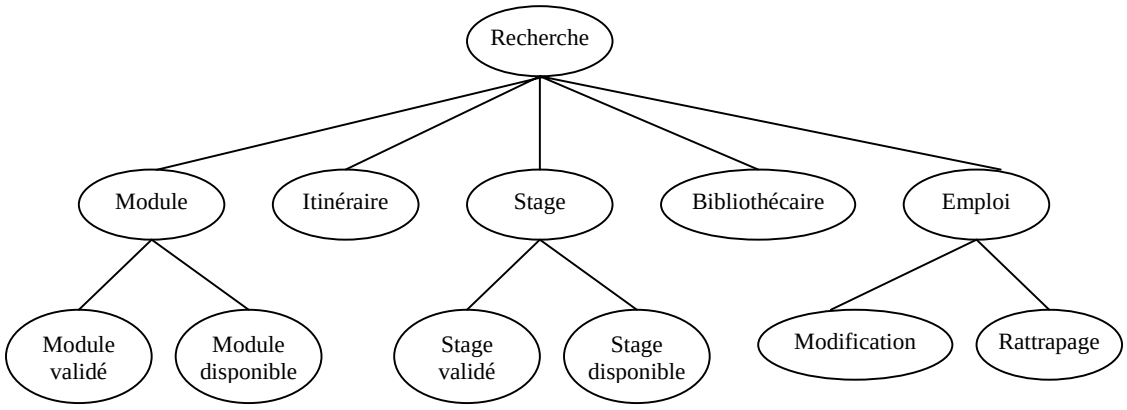


Figure 3.4. Organisation hiérarchique des modèles d'agent de recherche.

La Figure 4.5. décrit une organisation hiérarchique des modèles d'agent de recherche d'information. Un ou plusieurs agents peuvent être crées par modèle d'agent. Cette organisation hiérarchique des modèles d'agents ne reflète d'aucune manière l'organisation physique des agents issus de ces modèles. Tout dépend du modèle d'organisation effectif intégrés aux agents. Cependant cette organisation hiérarchique renseigne sur les domaines d'activités dont les agents issus de ces modèles appartiennent. Il suffit seulement de ne considérer que la racine de l'arbre obtenu puisque tous les nœuds fils correspondent nécessairement à des modèles d'agent spécialisés.

En effet, si des agents issus des modèles d'agent de recherche d'information (voir Figure 4.5.) doivent coopérer avec des agents issus des modèles d'agent de gestion de profil (voir Figure 4 .6.), la coordination entre ces agents se fera au niveau de l'agent de coordination.

La Figure 4.7. illustre la coordination entre les agents de recherche d'information et ceux de gestion de profil de l'utilisateur.

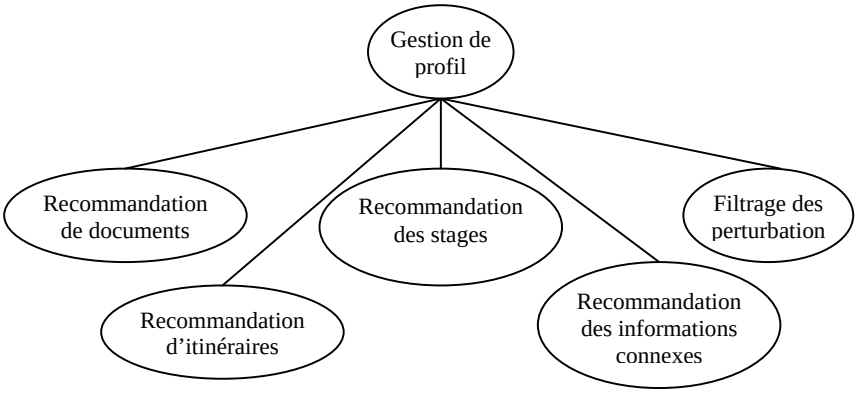


Figure 3.5. Organisation hiérarchique des modèles d'agent de gestion de profil.

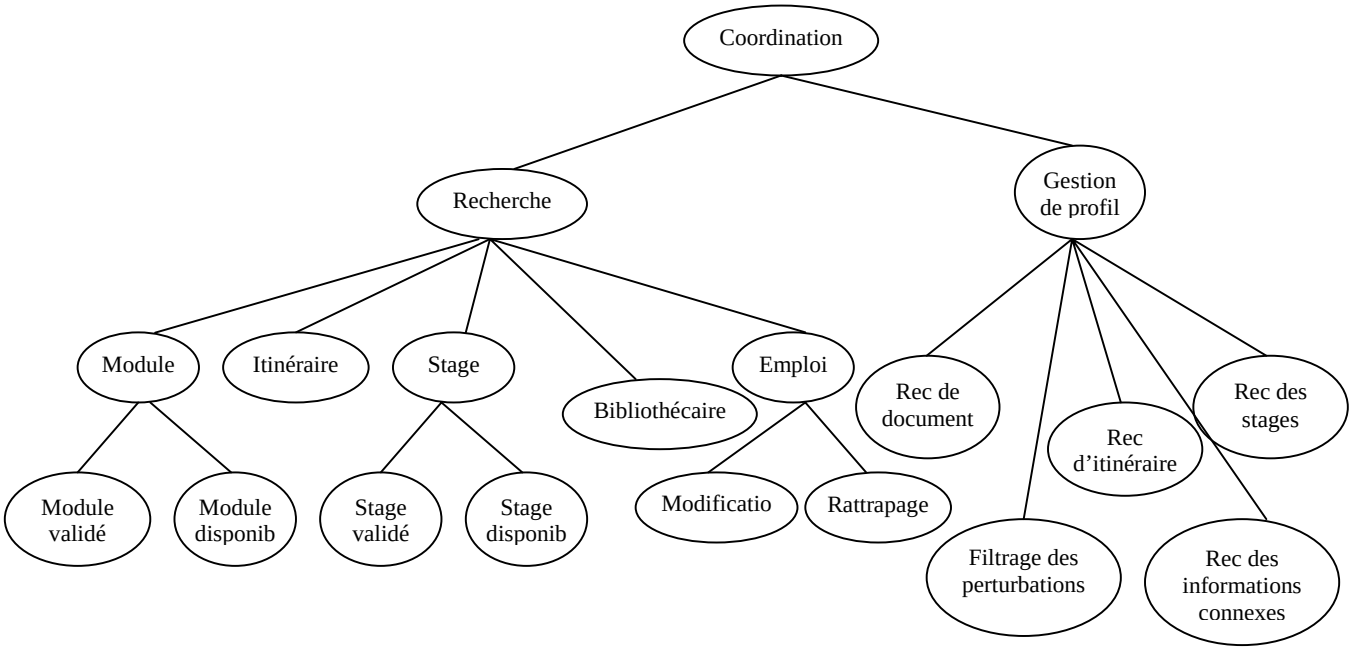


Figure 3.6. Coordination des agents de domaines d'activités différentes.

Coordination des messages échangés entre l'application externe et les agents applicatifs

Les messages échangés entre l'application externe et les agents applicatifs transitent par l'agent de coordination. Bien évidemment, une transformation de ces messages est effectuée (par l'agent de communication, qui sera décrit par la suite) pour que les messages envoyés par l'application externe soient compréhensibles par les agents logiciels et vice versa. C'est l'agent de coordination qui distribue les messages aux agents applicatifs concernés et renvoie les réponses aux applications externes [Anli .06].

Le diagramme d'activité de la Figure 4.8. donne un exemple d'échange de message entre une application externe (un serveur web) et les agents de PerESPERE.

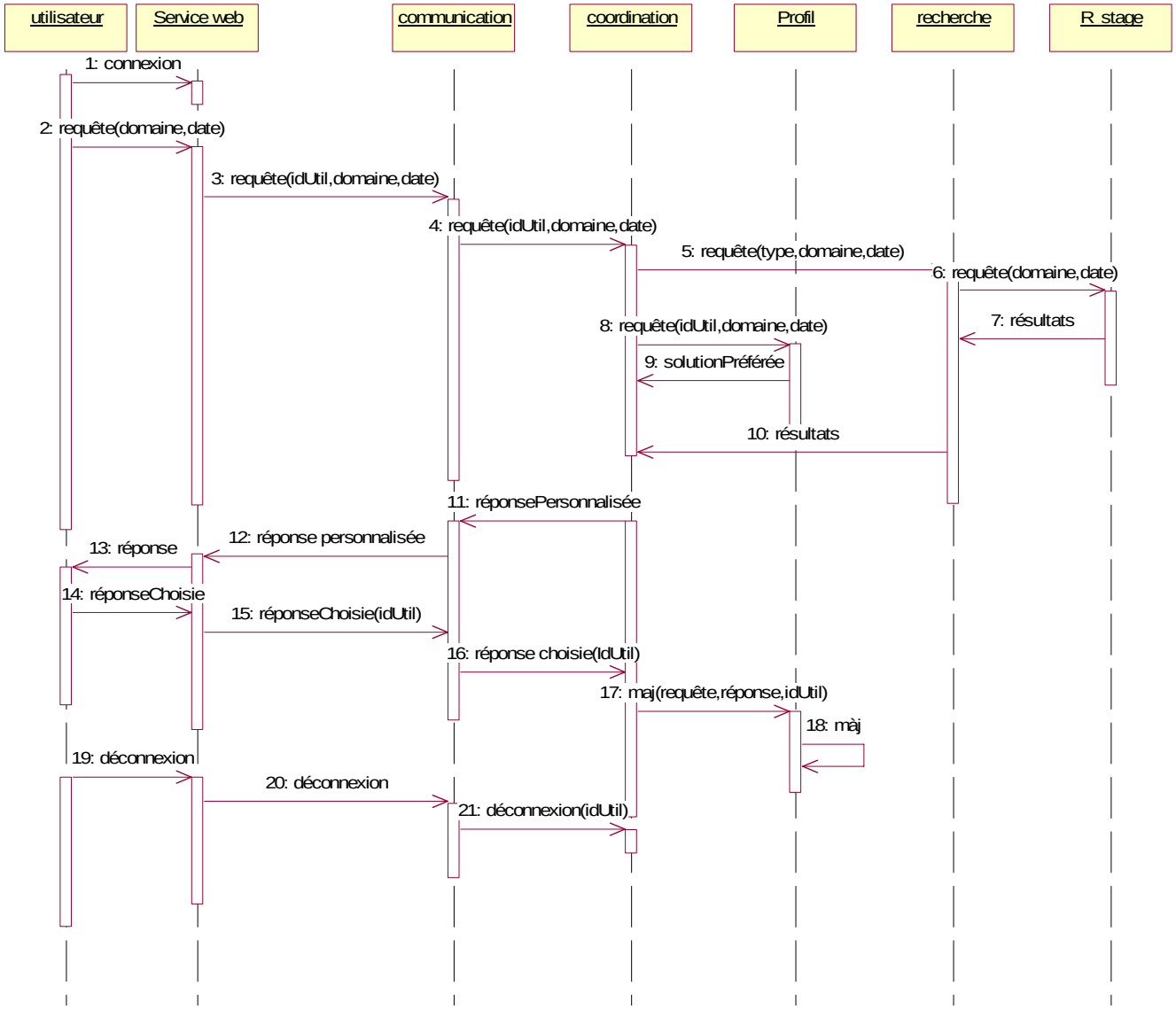


Figure 3.7 Coordination des messages entre l'application externe et les agents applicatifs.

3.2.2. L'agent de communication

L'agent de communication permet la traduction des requêtes envoyées par les applications externes en des messages compréhensibles par l'agent de coordination et vice versa. Cet agent diffère des autres agents de PerESPERE puisqu'il n'est pas un agent «Magique pur». En effet, même si cet agent intègre des aspects de Magique (il utilise l'API Magique pour envoyer les messages à l'agent de coordination), son fonctionnement et sa structure interne sont différents de ceux d'un agent Magique. La Figure 4.11. présente l'architecture de l'agent de communication et ses interactions avec les applications externes et avec l'agent de coordination. L'agent de coordination se décompose en deux parties : la première partie appelée *ApplicationCommunicator* se charge de la communication avec les applications externes et la deuxième partie appelée *MagiqueCommunicator* s'occupe de l'interaction avec l'agent de coordination[Anli .06].

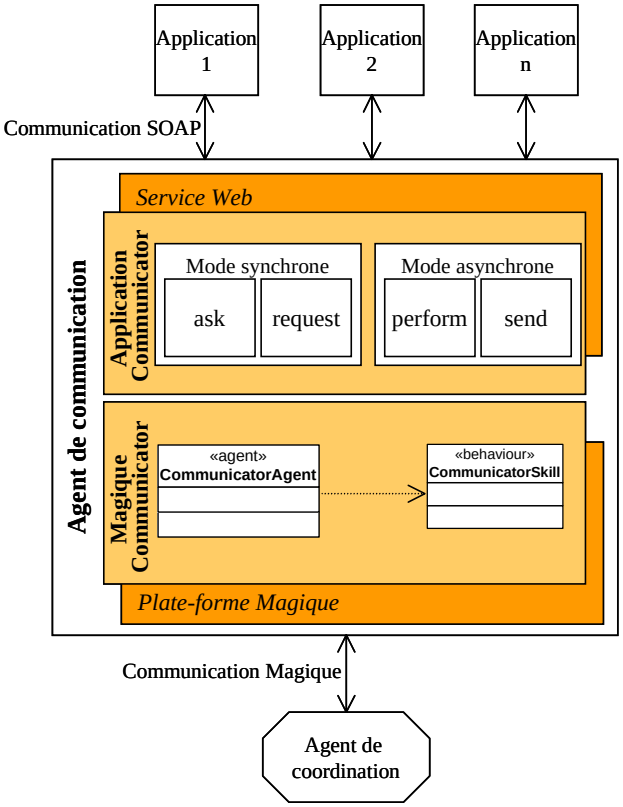


Figure 3.8. Architecture de l'agent de communication.

La partie ApplicationCommunicator

La partie ApplicationCommunicator utilise le protocole SOAP (Simple Object Access Protocol) [Kadima et Monfort 03] pour l'interaction avec les applications externes qui peuvent être écrites avec n'importe quel langage de programmation. Elle fournit quatre primitives de communication (*ask*, *request*, *perform* et *send*) exposés sous forme de services web. Les paramètres d'entrée/sortie de ces services sont des chaînes de caractères. En effet, actuellement, il n'existe pas d'équivalence systématique entre les types

d'objet des différents langage de programmation. Un objet XML Java, par exemple, ne sera pas reconnu lors de la communication SOAP comme un objet XML C#. Seuls les types de bases (entier, réel, chaîne de caractère, etc.) sont relativement bien reconnus par la majorité des langages de programmation. Pour cela, nous avons opté pour les chaînes de caractères sachant que le but, ici, est seulement de transférer des messages (donc pouvant être encodés sous forme de chaînes de caractères). Ces messages peuvent être structurés sous le format XML. La Figure 4.12. donne un exemple de message pouvant être envoyé par une application externe. Dans cet exemple, l'application externe fournit à PerEspere le « *distinguished name* » pour référencer un objet (dans cet exemple, c'est pour identifier un utilisateur) enregistré dans un annuaire LDAP (Lightweight Directory Access Protocol) [Rizcallah 00] et une requête de recherche d'itinéraire comportant les lieux de départ et d'arrivée ainsi que la date et l'heure de départ.

```
-<Entry>
      -<DN>
      LDAP://SRV:389/CN=test,OU=Sitp,DC=DOMESPERE,DC=local
      </DN>
      -<Request>
            <LieuDepart>Centre Ville</LieuDepart>
            <LieuArrivee>ENIS</LieuArrivee>
            <HeureDepart>08:00</HeureDepart>
            <Date>1/1/2006</Date>
      </Request>
</Entry>
```

Figure 3.9. Exemple de message envoyé par une application.externe à l'agent de communication

Deux des quatre primitives permettent une communication synchrone et les deux autres permettent une communication asynchrone.

- **Communication synchrone :** l'application appelante attend une réponse de la part de PerESPERE pour poursuivre son exécution. Les interfaces fournies sont :

- public String ask(String questionName); cette primitive peut être utilisée par l'application externe lorsqu'elle veut poser une question. La question est spécifiée au travers du paramètre d'entrée « questionName ». Par exemple, supposons qu'un agent de PerEsperer fournit un service « getUsers » permettant de renvoyer la liste des utilisateurs inscrits au système. Les applications externes peuvent demander la liste des utilisateurs inscrits en utilisant la primitive de la manière suivante *result=object.ask("getUsers")*[10].

- public String request(String serviceName, String param); cette primitive assure les mêmes fonctionnalités que la précédente sauf qu'elle autorise l'appel de services des agents comportant un paramètre d'entrée. Par exemple, l'instruction *result=object.request ("searchItinerary",param);* avec *param* une variable contenant la chaîne de caractères de la Figure 4.12. **Figure** 3.9. **Exemple de message envoyé par une application.externe à l'agent de communication**

- par exemple, permet à une application de solliciter le service "searchItinerary" d'un agent en lui passant le paramètre *param*. Bien évidemment, l'agent de communication transmet d'abord le message à l'agent de coordination qui va se charger d'envoyer la requête à l'agent adéquat.

[10] La syntaxe varie selon le langage de programmation utilisé. Dans cet exemple c'est en Java.

- **Communication asynchrone :** l'application appelante n'attend pas de réponse de la part de PerEspere. Elle envoie l'information et poursuit son exécution. Les interfaces fournies sont :

  - public void perform(String eventName); cette primitive peut être utilisée par l'application externe pour déclencher un événement à un agent de PerEspere sans attendre de réponse en retour. Par exemple, réveiller des agents, signaler le lancement d'une application, signaler l'arrêt d'une application etc.

  - public void send(String msgName, String msg); cette primitive assure les mêmes fonctionnalités que la précédente sauf qu'elle est utilisée pour déclencher des événements nécessitant un paramètre d'entrée. Comme pour la primitive *request*, lorsqu'il y a plusieurs paramètres, ils peuvent être structurés sous le format XML. Envoyer le choix de l'utilisateur, informer la nature de la plate-forme d'interaction de l'utilisateur, envoyer la localisation de l'utilisateur, etc. sont des exemples de cas pouvant nécessiter l'utilisation de cette primitive.

Ces primitives permettent de récupérer les messages des applications externes pour les envoyer à la partie MagiqueCommunicator.

La partie MagiqueCommunicator

La partie MagiqueCommunicator récupère les messages issus de la partie ApplicationCommunicator pour les traduire en des requêtes compréhensibles par des agents Magique. Concrètement, elle instancie une plateforme Magique contenant un agent Magique nommé *CommunicatorAgent*. Cet agent a la capacité de se connecter à l'agent de coordination de PerEspere grâce la compétence *CommunicatorSkill* (voir Figure 4.13.) pour pouvoir communiquer avec lui.

Cette partie est fournie sous forme d'une API Java pouvant être importée dans une application Java indépendamment de PerEspere. Ainsi lorsqu'une application Java a besoin de communiquer avec un agent Magique, il suffit de créer un objet de la classe *MagiqueCommunicator* (voir Fgure 4.14.). L'application pourra ensuite invoquer des services d'agents Magique par invocation des méthodes de l'objet crée.

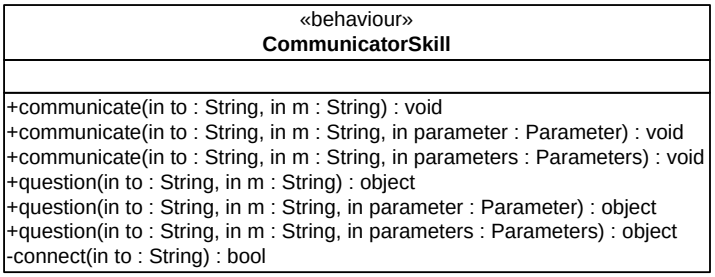


Figure 3.10. La compétence CommunicatorSkill.

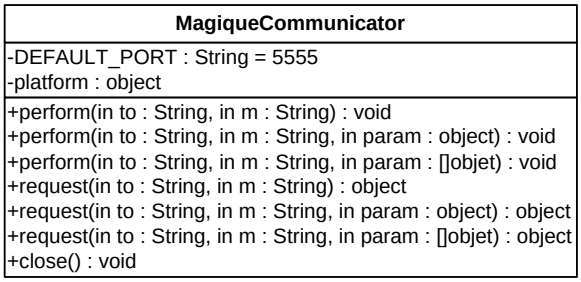


Figure 3.11. La classe MagiqueCommunicator.

3.2.3. L'agent d'administration

L'agent d'administration permet la gestion et l'administration de PerESPERE. Son rôle principal est de permettre à un utilisateur (admnistrateur) d'interagir avec les agents logiciels composant PerESPERE pour l'évolutivité. Cet outil se présente sous forme d'un agent Magique disposant des compétences nécessaires permettant l'administration et la gestion des agents Magique. La Figure 4.15 présente le diagramme de classes de l'agent d'administration de PerESPERE définit dans Persyst sous le nom de PSManager..

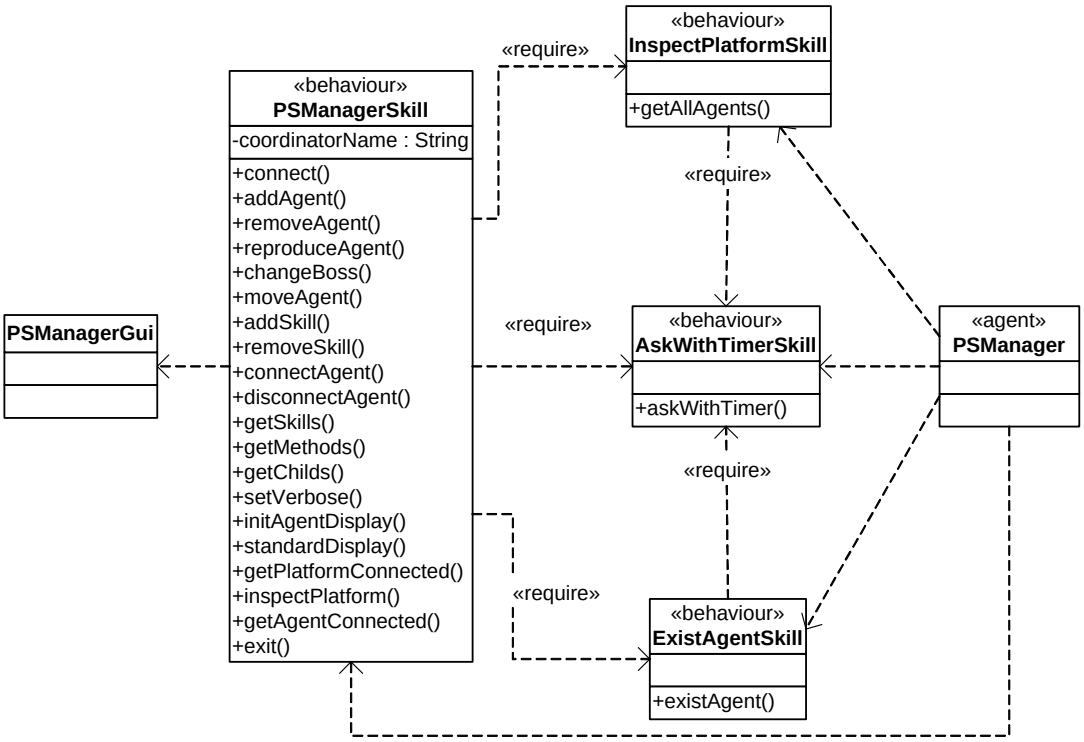
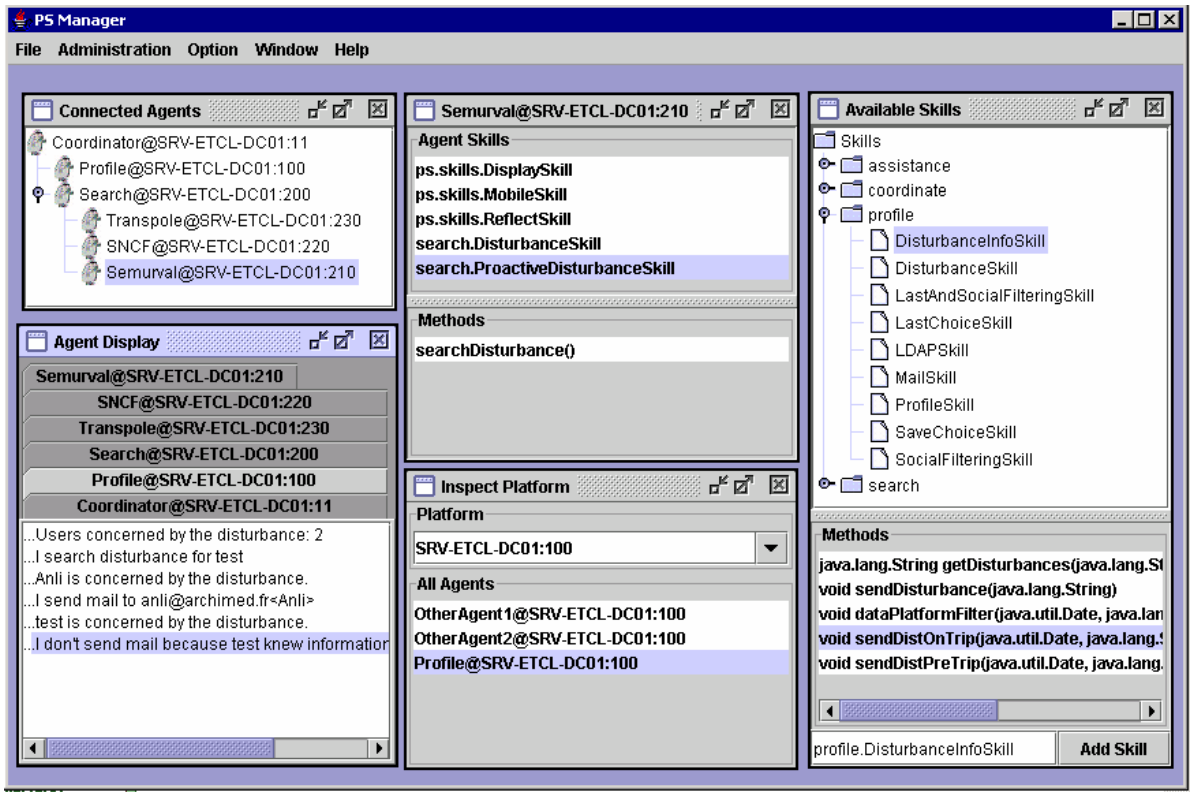


Figure 3.12. Le diagramme de classes de l'agent d'administration [Anli .06].

- *InspectPlatformSkill* permet à l'agent d'administration de connaître les agents présents dans une plate-forme donnée.



- *AskWithTimerSkill* permet d'envoyer des requêtes avec attente de réponse sans être bloqué indéfiniment au cas où il ne reçoit pas de réponse.
- *ExistAgentSkill* permet de savoir si tel agent existe.
- *PSManagerSkill* fournit des services pour la gestion des agents (création, suppression, reproduction d'un agent), pour la manipulation de leur topologie (organisation, distribution), pour l'administration de leurs compétences (ajout, suppression de compétences) et pour la (dé)connexion des agents (connexion, déconnexion).

Figure 3.13. L'agent d'administration de PerESPERE.

3.2.4. Les agents de recherche

Les agents de recherche se sont les agents applicatifs responsable de l'exécution des requêtes des utilisateurs pour trouver les solutions qui répondes à leurs requêtes.

La définition des agents de recherche dépend des services offerts par le système d'information à personnaliser. L'intégration de ces agents peut être fait au fur et à mesure que nous incluons des services à notre application. Pour répondre aux quatre services principales de notre système « Portail étudiant », nous définition les modèles d'agent de recherche suivants (voir Figure 4.5):

– Agent de recherche de modules : nous définissons deux modèles d'agent responsable de ce type de recherche : ceux pour les modules validés et pour les modules disponible.

- Agent de recherche d'itinéraires : ces agents aident l'étudiant à chercher et personnaliser ses itinéraires.
- Agent de recherche de stages : ces agent sont en recherche permanente de stages qui peuvent intéressés l'étudiant, en outre, ces agents permettent à l'étudiant à chercher les stages qui les à déjà validés et ceux qui sont disponibles et peuvent l'intéressés.
- Agent de recherche bibliothécaire : ces agent sont modélisés pour la recherche de documents bibliothécaires qui répondent aux requêtes des étudiants.
- Agent de recherche d'emploi et de rattrapage pour assurer la recherche des emplois et des éventuels rattrapages. Ces agents sont recherche permanente de modification des emplois et des rattrapages.

Les agents de recherche suivent une modélisation hiérarchique définie dans la figure 4.5. La coordination entre les différents modèles d'agent de recherche est assurée par l'agent de recherche (R) où l'échange de message (requête) passe essentiellement par ce dernier pour être par la suite transmit à un agent adéquat de recherche.

La compétence essentielle de ces agents est « SearchSkill » qui permet le lancement de l'exécution des requêtes des utilisateurs. En effet, l'exécution des requêtes se fait au niveau du système de médiation, mais ce sont les agents de recherche qui traduisent les requêtes de l'utilisateur pour pouvoir être exécuté au niveau du médiateur, en plus ces agents migrent au niveau du médiateur pour récupérer les résultats de la recherche.

3.2.5. Les agents de profil

Les agents de gestion de profil ce sont les agents responsables de l'élection de la solution pertinente pour l'utilisateur. En fait, ces agents se base sur l'historique de l'utilisateur et éventuellement l'historique des autres utilisateurs, pour trouver la solution adéquate et ignore les résultats qui se voient non intéressantes pour l'utilisateur.

Les principales compétences des agents de gestion de profil sont :

- *BayesFilteringSKiIl* permet de personnalisée la réponse du système en se basant sur le filtrage collaboratif à base des réseaux Bayésien.
- *LastChoiceSkill* cette compétence filtre les résultats pour retourner la solution choisie par l'utilisateur lors de la dernière exécution de la même requête.
- *SocilaFelteringSkill* permet de filtrer les résultats en se basant sur le filtrage social qui vise à établir une corrélation entre l'utilisateur actuel et les autres utilisateurs du système.
- *KSocilaFelteringSkill* cette compétence établie la corrélation entre les utilisateurs à un facteur K près.

La figure 4.17 récapitule les principales classes de l'agent de gestion de profil qui permettent d'identifier la solution pertinente en se basant sur une près authentification de l'utilisateur et une mise à jour automatique de son profil à chaque exécution e requête.

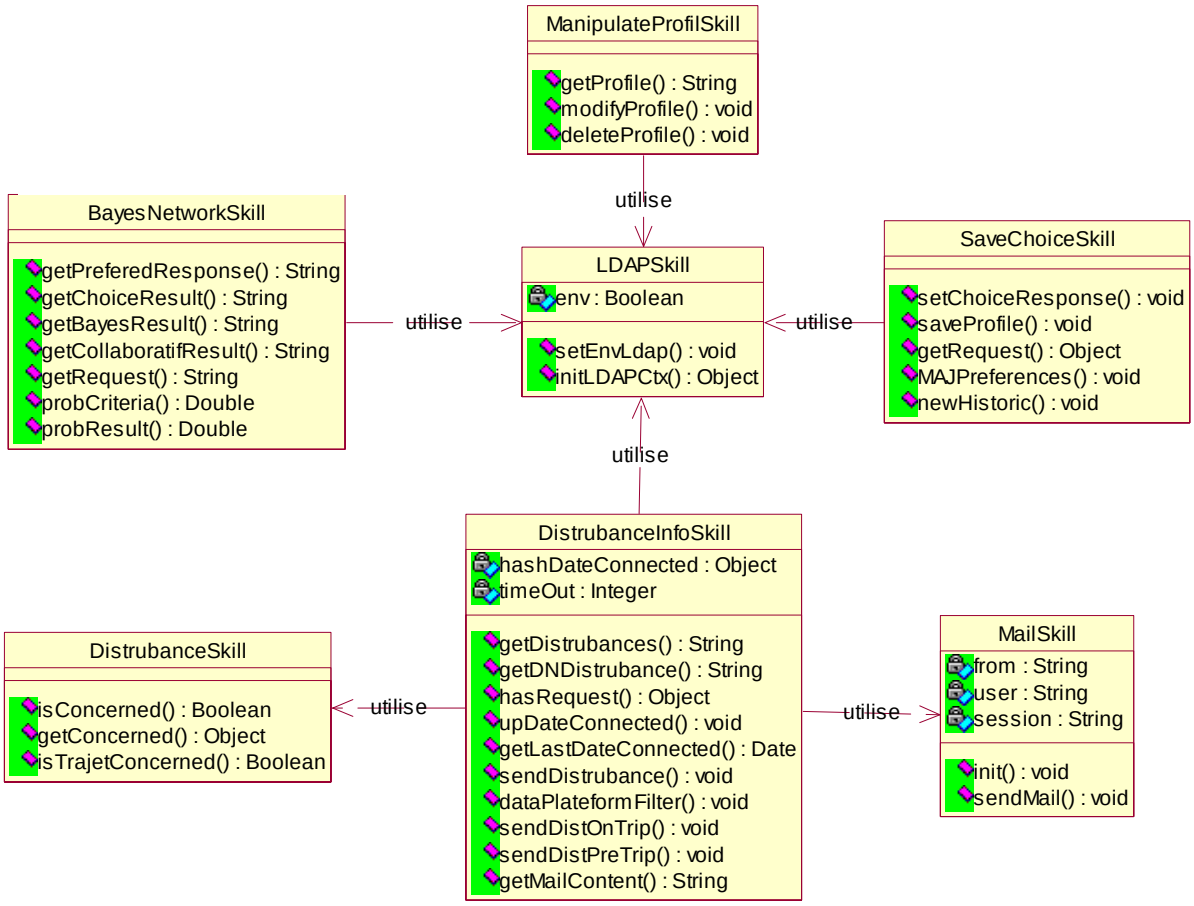


Figure 3.14. Diagramme de classe de l'agent profil.

Conclusion

Ce chapitre a proposé un système de personnalisation sur mesure pour personnaliser le système d'information ESPERE. Le système de personnalisation est appelé PerESPERE qui peut être utilisé pour le développement d'un système d'information personnalisé. PerESPERE est construit à partir d'une architecture multi-agents ce qui lui confère des caractéristiques comme l'adaptabilité, l'autonomie et l'assistance lui permettant de prendre en compte différents types de personnalisation.

PerESPERE distingue les *agents applicatifs* qui sont définis lors d'une intégration de PerESPERE dans un système d'information (comme les agents de gestion de profil utilisateur, de recherche d'information, d'assistance, etc.) *des agents systèmes* qui sont l'agent de communication, l'agent de coordination et l'agent d'administration. Ces trois agents systèmes forment le noyau de PerESPERE et chacun joue un rôle précis dont tous les trois combinés permettent à PerESPERE de répondre à ses objectifs de multi-applications, de distributivité et d'évolutivité.

Chapitre 4
MedESPERE : SYSTEME DE MEDIATION DES DONEES HETEROGENES

Sommaire

Introduction

Un des objectifs de nous travaux est de fournir un système de médiation permettant de surmonter le problème d'hétérogénéité et de distributivité des données.

Nous proposons une solution pour intégrer des systèmes d'information hétérogènes fondées sur la médiation pour résoudre les conflits sémantiques et syntaxiques, tout en préservant l'autonomie des sources de données

4.1. Concepts utilisés

4.1.1. Comparaison entre GAV et LAV

Les différents systèmes d'intégration d'informations à base de médiateurs se distinguent par la façon dont est établie la correspondance entre le schéma global et les schémas des sources de données à intégrer.

En fait, on distingue l'approche Global As Views (GAV) de l'approche Local As Views (LAV) (voir Figure 5.1.). L'approche GAV, qui provient du monde des bases de données fédérées, consiste à définir le schéma global en fonction des schémas des sources de données à intégrer. L'approche LAV est l'approche duale.

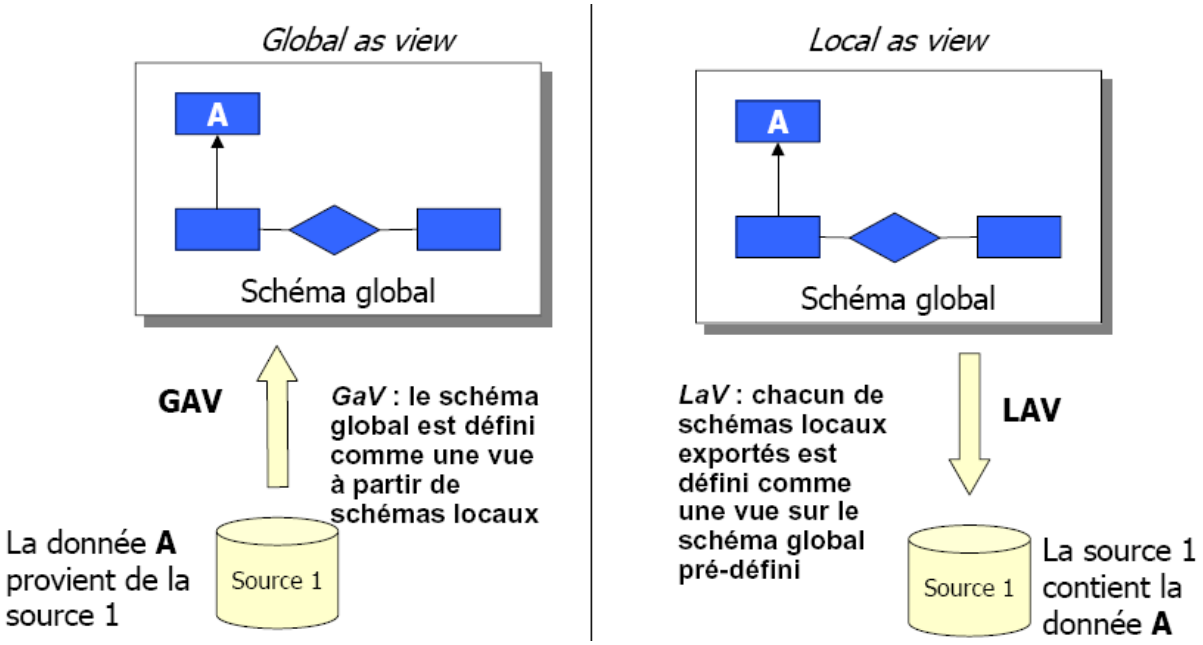


Figure 4.1. Les approches GAV et LAV.

Les avantages et inconvénients de ces deux approches sont inverses. Selon l'approche LAV, il est très facile d'ajouter une source d'information, cela n'a aucun effet sur le schéma global. En revanche, la construction des réponses à des requêtes est complexe, contrairement à la construction de réponses dans un système adoptant une approche GAV qui consiste simplement à remplacer les prédicats du schéma global de la requête par leur définition (voir tableau 5.1.).

	GAV	**LAV**
Qualité dépend de	comment les sources sont intégrées	comment les sources sont caractérisées
Ajout/suppression de source	m à j du schéma global	pas de modification du schéma global
Requête	Elaboration simple (unfolding)	Elaboration complexe (réécriture)

Tableau 4.1. Comparaison entre les deux approches GAV et LAV

Pour ce faire, nous nous appuierons sur un algorithme de réécriture dans une approche LAV (Local As View) proposé par [LeOR 96] pour notre médiateur « MedESPERE »

4.1.2. Rappels des principes de réécriture des requêtes

4.1.2.1. Exemple illustratif

Cette section décrit un exemple de système de médiation avec son schéma virtuel et les liens sémantiques qui le relient aux sources de données participantes, appelés aussi requêtes de médiation.

Pour la simplicité de l'exemple, nous nous limiterons à des sources de données relationnelles et nous nous plaçons dans un contexte LAV (Local As View). Dans la suite on utilisera S_v pour désigner le schéma virtuel et S_m pour l'ensemble de requêtes de médiation $\{S_1, \ldots, S_n\}$.

Exemple 1 : Schéma virtuel (S_v)
Ecole(idE, nom, adresse)
Etudiant(idEt,nom,prenom,age,handicap,abonnementTransport,adresse,idE,sction,groupe)
Doument(idD,idE,titre,auteur,theme,edition,numVersion)
Emploi(idEm, idE,section,groupe)

Notre exemple de système d'intégration de données traite des Ecole, des Etudiant, des documents et des Emplois. Son schéma virtuel est composé des relations de l'exemple 1. soient les sources de données suivantes :

- **EcoleEnis :** source des données solaires et documentaires de l'ENIS
- **EtudiantHandicap :** source contenant les étudiants handicapés

Chaque source est décrite par une requête de médiation exprimée à la Datalog. L'exemple 2 montre la définition de la source EcoleEnis. La partie gauche de la requête correspond au schéma de la source et la partie droite contient un atome pour chaque relation virtuelle invoquée (document, Emploi et Etudiant).

Exemple 2 : **Définition de la source**
S_1 : Ecole(idE, nom, adresse) :-
Ecole(idE, 'ENIS', adresse)
Etudiant(idEt,nom,prenom,age,handicap,abonnementTransport,adresse,idE,sction,groupe)
Doument(idD,idE,titre,auteur,theme,edition,numVersion)
Emploi (idEm, idE,section,groupe)

Les contraintes sur le contenu des sources sont exprimées en remplaçant un attribut par sa valeur. Par exemple, pour la définition de la source EcoleEnis, l'attribut nom est remplacé par la valeur 'ENIS' dans l'atome qui correspond à la relation virtuelle Ecole.

Les requêtes de médiation ainsi définies vont servir à la réécriture des requêtes utilisateurs.

4.1.2.2. Principe de réécriture des requêtes

Cette section rappelle les principes des mécanismes de réécriture et d'enrichissement de requêtes que nous avons retenus pour illustrer l'importance de la personnalisation dans un système d'intégration de données.

La réécriture d'une requête dans une approche Local As View consiste à déterminer les sources contributives pour l'exécution de la requête utilisateur [BaHM 04] [LeOR 96] et à utiliser leurs définitions pour reformuler cette requête. Il existe deux classes principales d'algorithmes de réécriture: les algorithmes de règles inversées et les algorithmes à base de tas (Bucket-based) [BaHM 04]. Pour notre étude, nous utilisons un algorithme de la dernière classe.

Il fonctionne en deux phases [CaLL 01] :

- Création d'un tas (bucket) pour chaque atome g de la requête Q qui contient les vues (requêtes de médiation) contributives pour cet atome. Ce sont les vues à partir desquelles on peut obtenir des tuples de l'atome g.

- Construire des réécritures candidates et ne garder que les réécritures qui sont incluses dans la requête. Chaque réécriture candidate est une requête conjonctive obtenue en prenant une vue de chaque tas. Une requête Qi est incluse dans une autre Qj si pour toute base de données D, l'ensemble des tuples retournés par l'évaluation de Qi sur D est un sous-ensemble des tuples retournés par Qj.

Prenons une requête utilisateur Q1, exprimée sur le schéma virtuel de l'exemple 1, qui cherche les étudiants handicapés:

Exemple 3 : **Requête initiale**
Q_1 : SELECT idEt , nom, prenom, age,E.nom, E.adresse, T.adresse FROM Ecole E, Etudiant T
WHERE T.handicap='OUI'

Soit R l'algorithme de réécriture de Q_1 sur les définitions de sources Sm = {S_1, …, S_n} (R(Q_1 / S_m)), la première phase de réécriture de Q_1 correspond à la construction des 2 tas de requêtes de médiations contributives, {S_1,…, S_p} et {S_p,… , S_n} respectivement pour les relations

virtuelles Ecole et Etudiant. La deuxième phase de R consiste à combiner les requêtes de médiation de chaque tas ; ce qui implique une réécriture RW_1 représentée par l'union de $C_p^1 * C_{n-p}^1$ requêtes conjonctives w_1^i correspondant aux possibilités de choix de sources contributives :

$$RW_1 = \bigcup_{\substack{i \\ j\in\{1,..,p\} \\ k\in\{p+1,...,n\}}} w_1^i /(S_j, S_k)$$

Où $w_1^i /\{S_j, S_k\}$ correspond à la i-ème réécriture de la requête Q_1 qui est faite avec les requêtes de médiation S_j et S_k.

Chacune des requêtes conjonctives qui composent la réécriture de Q1, délivre des informations dont la sémantique est différente en fonction des prédicats de sélection qu'elle vérifie.

4.2. Architecture de MedESPERE

Nous proposons « MedESPERE » une solution pour l'intégration de systèmes d'informations hétérogènes, fondée sur la médiation pour résoudre les conflits sémantiques et syntaxiques. Nous visons l'hétérogénéité sémantique aussi bien au niveau des schémas (structures) qu'au niveau du contenu (données), et ceci, tout en préservant l'autonomie des sources de données.

Notre système de médiation, MedESPERE comporte trois niveaux: le niveau interface, le niveau médiateur, le niveau adaptateur et le niveau des sources locales (figure .)[Benhlima et chiadmi.06]. Dans la suite, nous décrirons le médiateur et l'adaptateur ainsi que le traitement général d'une requête.

4.2.1. Médiateur

Au niveau médiateur, la requête passe d'abord par la phase de **réécriture**, puis par les phases d'optimisation, d'exécution et enfin la présentation des résultats (réponse). Outre les modules de traitement de la requête, le médiateur dispose d'un catalogue.

La requête représentée sous format XML passe par le module de réécriture utilisant le mapping pour connaître les sources locales. Puis, chaque concept global est remplacé par son correspondant local.

Les sous requêtes sont ensuite optimisées avant d'être exécutées puis intégrées selon le plan d'exécution généré à partir du mapping. Le résultat est enfin présenté en utilisant de nouveau le XML.

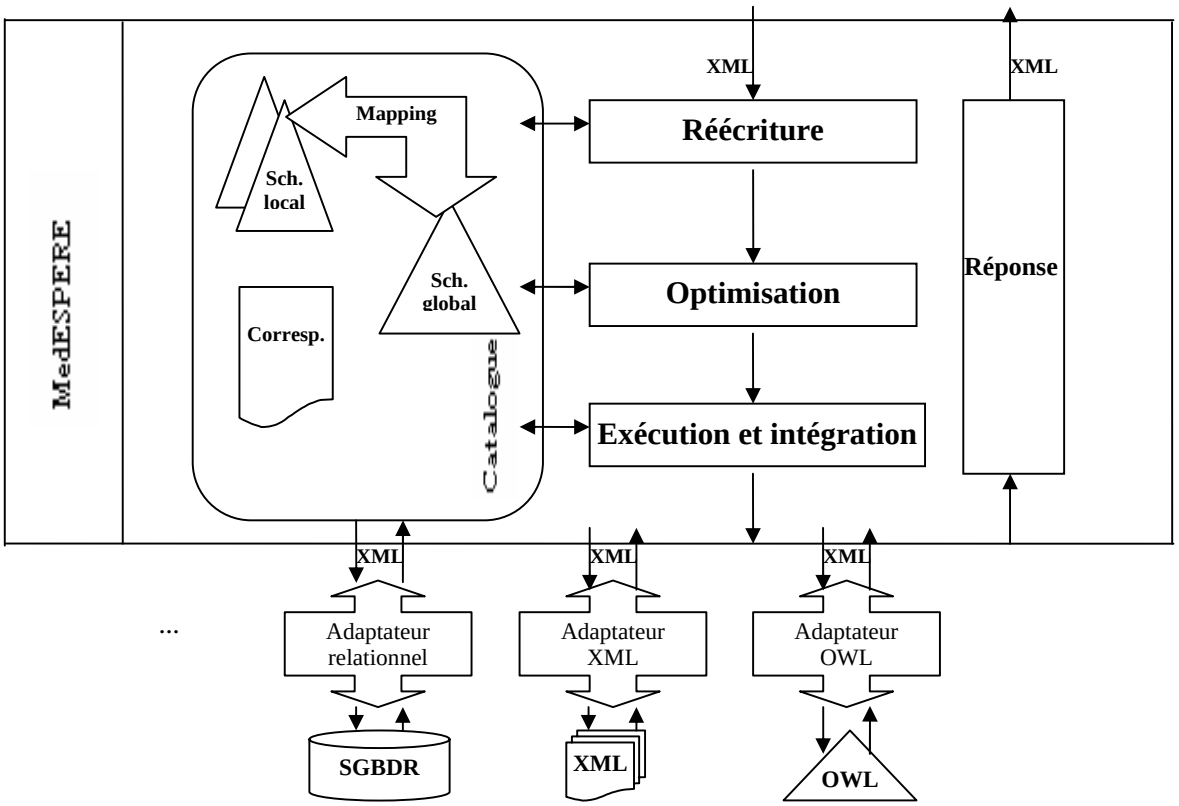


Figure 4.2. Architecture de MedESPERE.

4.2.2. Adaptateur

L'hétérogénéité syntaxique est résolue au niveau des adaptateurs. En effet, ces derniers adaptent la sous requête exprimée en XML au langage de requête de la source et envoient la requête à la source en utilisant le protocole adéquat. Les résultats renvoyés par la source sont alors transformés en XML avant d'être acheminés vers le médiateur.

4.3. Enrichissement sémantique

Avant de décrire plus en détail les composants qui contribuent à l'enrichissement sémantique, nous présentons un scénario d'utilisation simplifié pour illustrer l'hétérogénéité entre deux sources d'informations.

4.3.1. Scénario d'utilisation

Nous considérons ici un exemple d'application qui permet d'intégrer deux bibliothèques virtuelles. En plus des références sur les documents, chaque bibliothèque stocke les ouvrages sous format électronique. Un fragment des SI de chacune des deux bibliothèques est donné en figure 2 [Benhlima et chiadmi.06].

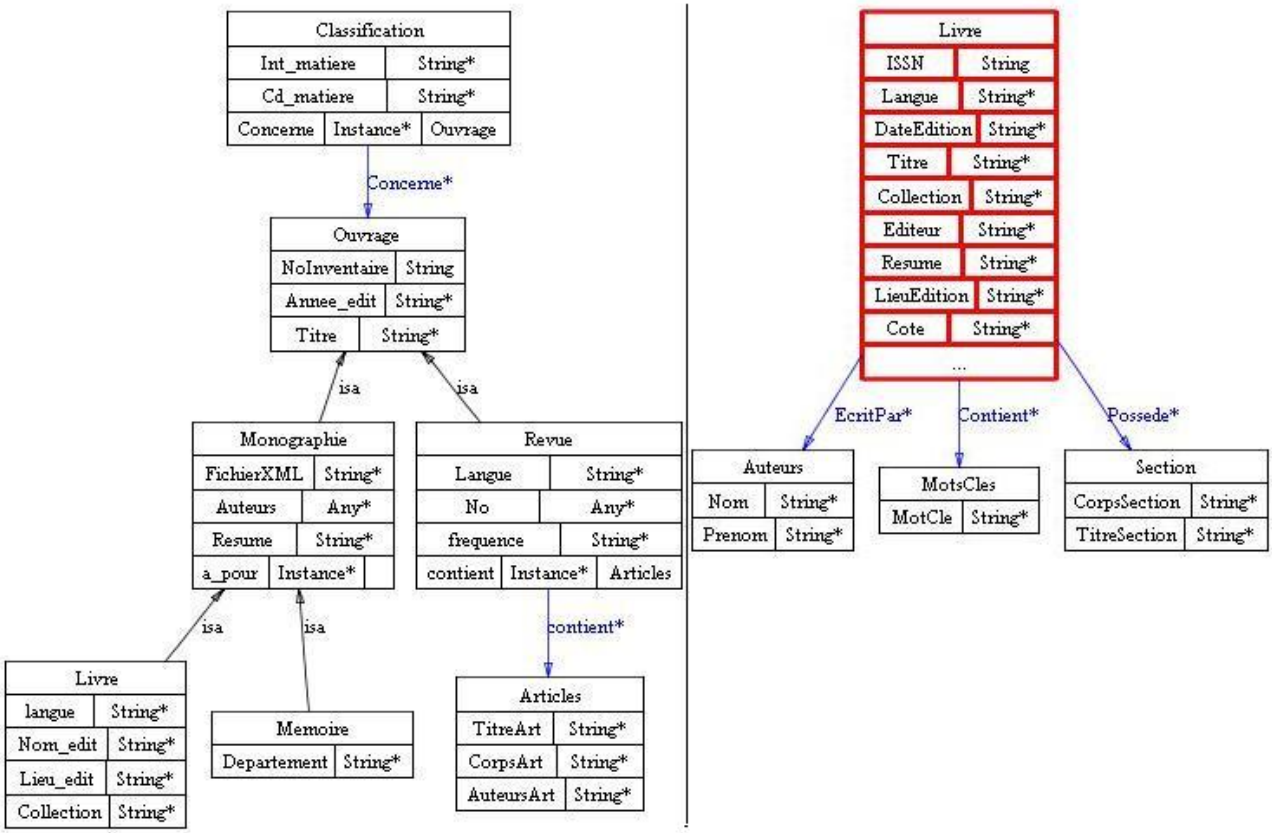


Figure 4.3. Fragment du système d'information des deux bibliothèques.

Les hétérogénéités entre les deux systèmes sont nombreuses ; nous citons à titre d'exemple les cas suivants :

- Dans le premier système, la classe Livre est une spécialisation de la classe Monographie, elle-même spécialisation de la classe Ouvrage. Dans le second système, la classe Livre regroupe tous les types d'ouvrages.
- L'auteur est un attribut dans le premier système alors que c'est une classe dans le second.
- Le premier système ne donne que l'année d'édition alors que le second donne la date d'édition complète.
- Pour le premier système, le contenu du document électronique est pointé par le nom de fichier, représenté par l'attribut FichierXML, d'une part et par la classe Articles, d'autre part. Quant au contenu du second système, il est représenté par la classe Section.

Dans la suite, nous considérons le cas d'une requête qui demande l'année d'édition, ainsi que le contenu des documents relatifs aux "bases de données réparties".

4.3.2. Composants pour l'enrichissement sémantique

Dans cette section, nous décrivons les composants utilisés dans MedESPERE pour enrichir sémantiquement la requête, à savoir le catalogue et les modules de la réécriture.

4.3.2.1. Les ontologies

Le terme « ontologie » est employé dans plusieurs domaines et de manières différentes. Dans [Gruber .95], les ontologies sont présentées par Gruber comme «spécifications explicites d'une conceptualisation ». Une conceptualisation se rapporte à un modèle abstrait, comment les gens pensent généralement à une vraie chose dans le monde, par exemple un produit. Les spécifications explicites signifient que les concepts et les relations du modèle abstrait ont été exprimées par des noms et des définitions explicites. Une ontologie donne le nom et les descriptions des entités d'un domaine spécifique en utilisant les attributs qui représentent le rapport entre ces entités. Elle fournit un vocabulaire pour représenter et communiquer la connaissance au sujet du domaine et un ensemble de correspondances contenant les termes du vocabulaire à un niveau conceptuel. Par conséquent, une ontologie pourrait être employée pour des tâches d'intégration de données en raison de son potentiel de description de la sémantique des sources d'informations et de résolution les problèmes d'hétérogénéité.

Donc, les ontologies peuvent être employées dans le processus d'intégration, elles permettent de décrire la sémantique des sources d'informations et d'expliciter leur contenu. Pour l'intégration des sources de données, les ontologies peuvent être employées pour l'identification et l'association des concepts sémantiquement correspondants. Cependant, dans plusieurs projets les ontologies assurent des tâches additionnelles.

4.3.2.2. La description du contenu des sources de données

Dans presque toutes les approches d'intégration basées sur les ontologies, ces dernières sont employées pour la description explicite de la sémantique des sources d'informations. Mais la manière d'utiliser ces ontologies peut être différente. Trois directions différentes sont identifiées : l'approche avec une simple ontologie, l'approche avec de multiples ontologies et l'approche hybride.

- ***Approche avec une seule ontologie***

Cette approche utilise une ontologie globale qui fournit un vocabulaire partagé pour la spécification de la sémantique des sources de données (voir Figure 4.4). Toutes les sources de données sont liées à une ontologie globale. Chaque source est simplement liée à l'ontologie globale du domaine.

L'ontologie globale peut également être une combinaison de plusieurs ontologies spécialisées. Une des raisons de la combinaison de plusieurs ontologies peut être la modularisation du grand potentiel de cette ontologie. La combinaison est soutenue par desformalismes de représentation d'ontologie.

Cette approche présente un inconvénient majeur lors de l'ajout ou suppression de sources de données. En effet, la conceptualisation du domaine représenté dans l'ontologie peut nécessiter des changements. Cela a mené au développement des approches avec de multiples ontologies.

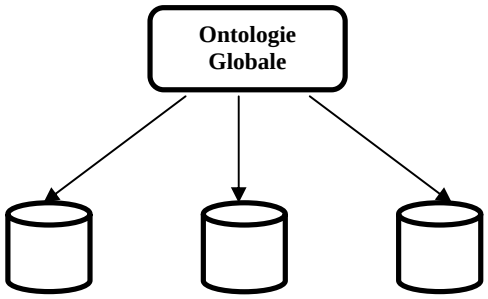


Figure 4.4. Approche avec une seule ontologie.

- ***Approche avec plusieurs ontologies***

Dans l'approche avec plusieurs ontologies, chaque source est décrite par sa propre ontologie (voir Figure 4.5). En principe, l'ontologie de chaque source peut être une combinaison de plusieurs autres ontologies mais on ne peut pas supposer que les différentes ontologies de source partagent le même vocabulaire.

L'avantage des approches avec plusieurs ontologies est que l'ontologie n'a aucun besoin d'engagement commun et minimal envers une ontologie globale. Chaque « ontologie de source » peut être développée sans le besoin de respecter ou connaître les autres sources et leurs ontologies. Cette architecture peut simplifier nettement la tâche d'intégration et soutient le changement (ajouter et enlever des sources). Cependant, le manque de vocabulaire commun rend difficile la comparaison entre les différentes « ontologies de source ». Pour pallier ce problème, un formalisme de représentation additionnel définissant le mapping entre ontologies est nécessaire. Le mapping entre ontologies identifie sémantiquement la correspondance des termes des différentes ontologies. Par exemple, si les termes sont sémantiquement égaux ou semblables. Mais ce mapping considère aussi les différentes vues sur le domaine. Par exemple le niveau d'agrégation et de granularité des différents concepts de l'ontologie. Cependant, Le mapping entre ontologies apparaît comme un processus délicat dans la pratique

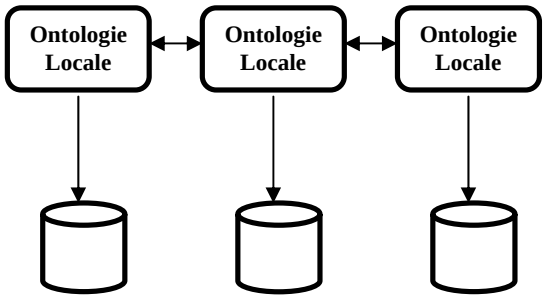


Figure 4.5. Approche basée su plusieurs ontologies.

- ***Approche hybride***

Pour surmonter les inconvénients des deux premières approches, les approches hybrides ont été développées (voir Figure 4.6). Cette approche décrit la sémantique de chaque source par sa propre ontologie comme avec l'approche à plusieurs ontologies. Mais afin de rendre les ontologies locales comparables entre elles, celles-ci sont construites à partir d'un vocabulaire partagé global. Ce dernier contient les termes de base (des primitives) d'un domaine qui peuvent

être combinés avec les ontologies locales afin de décrire une sémantique plus complexe. Parfois le vocabulaire partagé peut être une ontologie [Ruzzi .04].

Un des points intéressants dans l'approche hybride concerne la manière dont les ontologies locales sont décrites. Les termes pour un contexte proviennent d'une ontologie globale du domaine et des données elles-mêmes.

L'avantage de l'approche hybride réside dans le fait que de nouvelles sources peuvent facilement être ajoutées sans besoin de modification. Aussi, cette approche supporte l'acquisition et l'évolution des ontologies. L'utilisation d'un vocabulaire partagé rend les ontologies de source comparables et évite les inconvénients des approches avec de multiples ontologies. Mais l'inconvénient majeur des approches hybrides est que les ontologies existantes ne peuvent pas facilement être réutilisées, mais doivent être reconstruites à partir de zéro [Gruter.95].

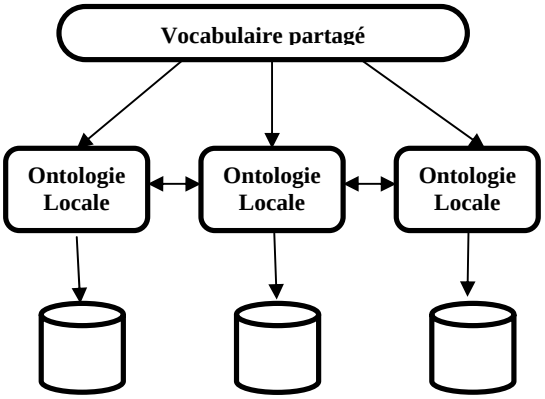


Figure 4.6. Approche hybride pour la description des sources de données.

4.3.2.3. Le développement des ontologies dans les systèmes d'intégration

L'approche que nous avons suivie [SETTOUTI .05] consiste à créer les ontologies locales, et à extraire l'ontologie globale à partir des différents concepts utilisés dans les ontologies locales. Trois étapes principales constituent cette méthode (voir Figure 4.8):

- Etape 1 : construction des ontologies locales;
- Etape 2 : construction d'un vocabulaire partagé sous forme d'une ontologie globale;
- Etape 3 : définition des différentes correspondances entre les sources, les ontologies locales et l'ontologie globale.

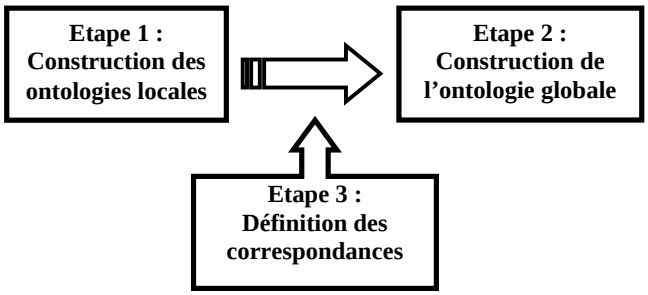


Figure 4.7. Méthode de construction des ontologies.

- ***Construction des ontologies locales***

Pour construire les ontologie locales Settouti propose deux étapes principales : analyse des source de données et définition des ontologies locales. La première étape est une analyse complète de chaque source indépendamment. Cette analyse consiste à rechercher les termes utilisés dans la source. Après cette analyse, la deuxième étape peut être lancée.

- ***Construction de l'ontologie globale***

La construction du vocabulaire partagé contient deux étapes principales : (1) analyse des ontologies locales ; (2) sélection de tous les concepts et résolution des problèmes d'hétérogénéité sémantique.

1. La première étape implique une analyse complète des ontologies déjà construites indépendamment.
2. Dans la deuxième étape, il faut localiser les problèmes d'hétérogénéité sémantique.

- ***Correspondances entre l'ontologie globale et les ontologies locales***

L'ontologie globale est construite à partir des ontologies locales. Pour identifier la source ontologique originelle des concepts dans l'ontologie globale, nous utilisons des annotations. En effet, le langage OWL permet d'annoter des concepts et des propriétés selon un schéma de méta-données prédéfini. **Settouti** annote les concepts de l'ontologie globale avec un schéma d'annotation qui contient trois propriétés :

1. Ontologie locale originelle du concept à annoter (Uri_ Local_Ontology)
2. L'agent responsable de cette ontologie locale (Agent_Local_Ontology)
3. Le nom du concept dans l'ontologie locale (Concept_Local_Ontology)

4.3.2.4. Réécriture

Il s'agit de décomposer la requête en sous requêtes à la fois mono source et sémantiquement enrichies. Rappelons que nous avons adopté l'approche LAV pour la définition du schéma globale. Ainsi, les vues sur les sources locales sont décrites en fonction des vues globales. De ce fait, un processus de réécriture est nécessaire pour exprimer la requête générée au départ en fonction des vues globales en requêtes destinées aux sources locales. Ce module va donc consulter le mapping du catalogue afin de remplacer la vue globale par ses vues locales.

Le module de réécriture prend en entrée la requête représentée sous forme XML. En sortie de ce module, un sous arbre constitué d'opérateurs algébriques est construit pour chaque sous requête. Les nœuds du sous arbre ne concernent que les concepts relatifs à la source. En effet, dans le cas où un nœud ferait intervenir des concepts existant dans plusieurs sources (cas d'un nœud select avec plusieurs conditions), le nœud sera éclaté en plusieurs nœuds pour ne garder, dans chaque noeud, que les concepts concernant la source. Dans le cas où le paramètre d'un nœud n'aurait pas de concept correspondant dans une source, le nœud sera écarté pour la source concernée.

4.4. Application de conversion (XML/Base de données)

4.4.1. Spécification et conception

4.4.1.1. Diagramme de cas d'utilisations

Cette application est composée essentiellement de deux parties :

- Convertir une base de données en XML
- Convertir l'XML en base de données

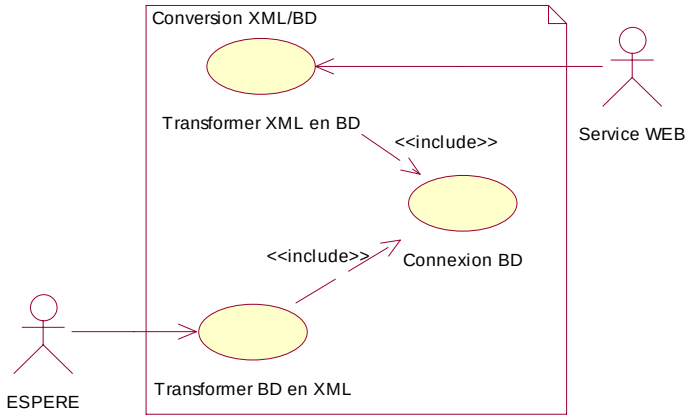


Figure 4.8. Diagramme des cas d'utilisations.

4.4.1.2. Description des principaux cas d'utilisation

❖ Cas d'utilisation << Transformer BD en XML >>

Use case *« Transformer BD en XML »*
Pré-conditions :
▪ ESPERE demande la conversion de BD en XML. ▪ ESPERE envoie l'adresse de la base de données. ▪ L'application se connecte à la base de données. ▪ L'application transforme les enregistrements de la base de données en champs XML. ▪ Elle enregistre le nouveau fichier XML portant le nom de la base et envoie une copie vers le système ESPERE.
Exception *:* ▪ Impossible de ce connecter à la base de données.
Extension *:*

❖ Cas d'utilisation << Transformer XML en BD >>

Use case « *Transformer XML en BD* »
Pré-conditions : ▪ ESPERE récupère un fichier XML du Service web.
▪ ESPERE demande la conversion de XML en BD. ▪ ESPERE le type de la base de données demandée et précise l'adresse de la base de données pour l'enregistrement. ▪ ESPERE envoie une copie du fichier XML. ▪ L'application transforme les champs XML en Base de données. ▪ Elle se connecte à la base de données. ▪ Elle enregistre les enregistrement créer dans la base de données concernée.
Exception *:* ▪ Impossible de ce connecter à la base de données.
Extension *:*

4.4.1.3. Diagramme de classes et de séquences

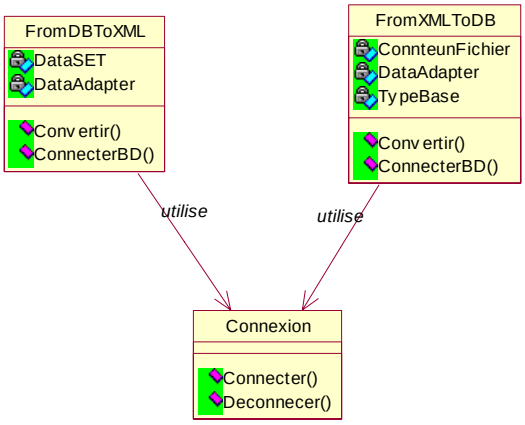


Figure 4.9. Diagramme de classes.

Description des méthodes principales :

- convertir () : méthode invoquée par le Système ESPERE pour convertir une base de données en XML et vise vers ça.
- Connecter () : méthode invoquer par l'application pour se connecter à une base de données proposée par le système ESPERE

Diagramme de séquence :

- *Cas d'utilisation << Transformer BD en XML >>*

Ce diagramme de séquence permet la transformation d'une base de donnée en fichier XML.

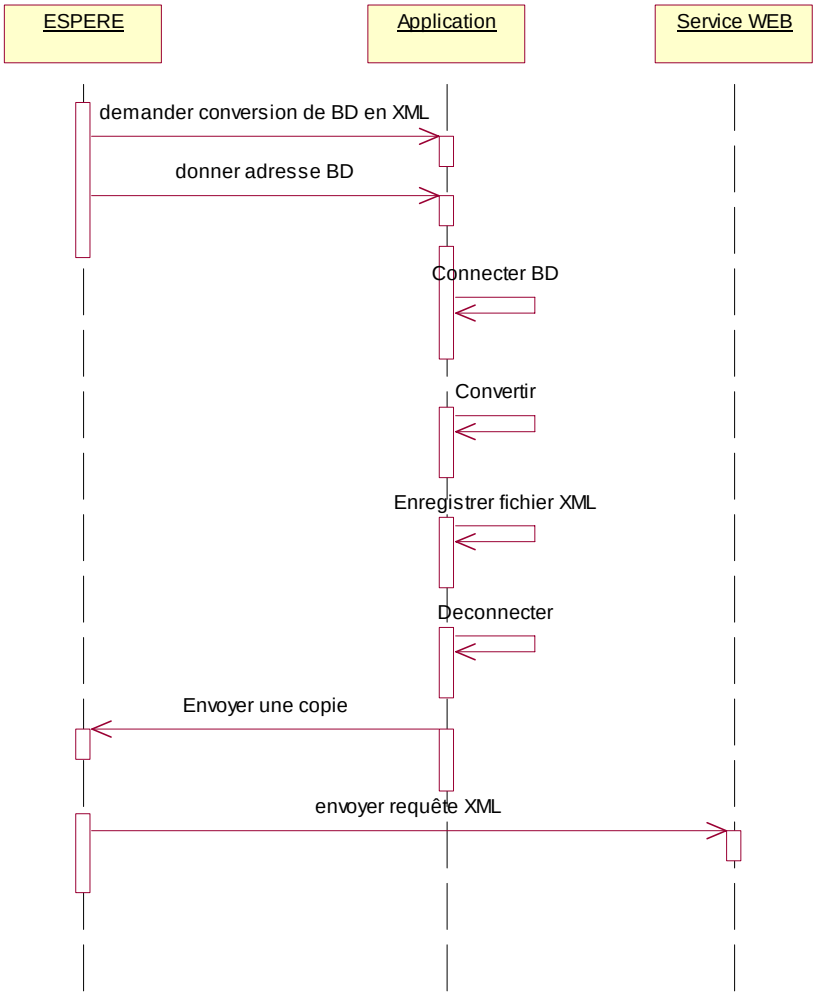


Figure 4.10. Diagramme de séquence « Transformer BD en XML ».

4.4.2. Réalisation

Nous prenons pour la clairette de l'exécution un exemple simple de base de données. En fait, la figure 5.5 représente la table « Etudiant » qui permet l'authentification des étudiant de notre système ESPERE. L'appel de l'application de conversion donne naissance au fichier « Etudiant.xml » présenté dans la figure 5.6. qui sera ensuite utilisé pour authentifier les étudiant enregistrés dans le système ESPERE via le système de personnalisation PerEspere.

Etudiant : Table

id	nom	prenom	age	handicap	abonTransport	domEtude	domParent	lieuEtude	telephone
8171777	Gassara	Yousra	24	☐	☑	sfax	sfax	sfax	95256868
8346874	NomEtudiant	etudiant2	21	☐	☐	tunis	sfax	tunis	23620903

Enr : 1 sur 2

Figure 4.11. Table « Etudiant ».

```
<?xml version="1.0" standalone="yes" ?>
- <Ensemble_x0020_des_x0020_contacts>
  - <etudiant>
      <id>8171777</id>
      <nom>Gassara</nom>
      <prenom>Yousra</prenom>
      <age>24</age>
      <handicap>false</handicap>
      <abonTransport>true</abonTransport>
      <domEtude>sfax</domEtude>
      <domParent>sfax</domParent>
      <lieuEtude>sfax</lieuEtude>
      <telephone>95256868</telephone>
      <email>gassara_yousra@voila.fr</email>
    </etudiant>
  - <etudiant>
      <id>8346874</id>
      <nom>NomEtudiant</nom>
      <prenom>etudiant2</prenom>
      <age>21</age>
      <handicap>false</handicap>
      <abonTransport>false</abonTransport>
      <domEtude>tunis</domEtude>
      <domParent>sfax</domParent>
      <lieuEtude>tunis</lieuEtude>
      <telephone>23620903</telephone>
      <email>etudiant2@yahoo.fr</email>
    </etudiant>
```

Figure 4.12. Fichier « Etudiant.xml ».

Conclusion

Nous avons proposé la conception d'un système de médiation pour l'intégration de sources hétérogènes et réparties. L'hétérogénéité sémantique (schéma et données) est traitée au niveau du médiateur. La résolution de conflits schématiques est réalisée grâce à la mise en correspondance entre les concepts du schéma global et ceux des schémas locaux représentant les sources hétérogènes. Ainsi, la requête est enrichie par des synonymes, des hyponymes et des hyperonymes pour avoir des réponses maximales.

Chapitre 5
Conception et réalisation du système ESPERE

Sommaire

Introduction

Ce chapitre est composé de trois parties.

Pour la première partie, nous y dégageons les besoins techniques et nous présentons les modèles des cas d'utilisations ainsi que les modèles d'analyse des cas d'utilisation.

Dans la deuxième partie, nous présentons la phase la plus importante dans le cycle de développement du Système d'information : c'est la phase d'analyse et de conception. Le support de cette phase par des techniques et des outils appropriés est crucial pour réussir un haut degré de fiabilité du système à développer tout en minimisant l'effort de développement. Pour ce faire, le choix de la méthode de conception UML se présente comme la solution la plus adéquate du simple fait que le processus de développement est piloté par les scénarios d'UML.

Tandis que dans la troisième partie nous allons présenter les différents environnements de travail que nous avons utilisé pour la modélisation, la conception et la réalisation du système ESPERE.

En outre, quelques concepts qui sont utilisés dans notre application se trouvent en annexe de ce rapport.

5.1. Spécification

Avant de développer une application, il est nécessaire d'analyser les exigences du système : La première étape du développement d'un projet. La détermination et la compréhension des besoins sont souvent difficiles étant donné que le développeur n'est pas forcement un connaisseur du domaine, ainsi que les informations recueillies lors de la phase de collecte d'informations devraient être filtrées afin de ne garder que celles nécessaires, pour ce fait et afin de bien mener le processus du développement on doit commencer par la modélisation des besoins des utilisateurs du système.

La modélisation des besoins peut être réaliser par une multitude de diagrammes ; Le processus de développement utilisé, tout en adoptant les notations UML, traduit cette nécessité par les diagrammes des uses-cases. Une fois identifiés et structurés, ces besoins devraient identifier les fonctionnalités du système.

Tout au long de la spécification et la conception de notre système ESPERE, en désigne par PerESPERE, le système de personnalisation « PerESPERE » proprement dit, en lui intégrant le système de médiation « MedEPSPERE ». Cette désignation est utilisée pour assurer la simplicité de nos diagrammes. e

5.1.1. Capture des besoins fonctionnels

Dans cette étape nous décrivons les principales fonctionnalités offertes par le système à développer en regroupant celles qui coopèrent en packages.

5.1.1.1. Identification des fonctionnalités

Les fonctionnalités à retenir lors de la conception d'un Espace de Services PERsonnalisés pour l'Etudiant peuvent se résumer comme suit :

- Le développement d'un démonstrateur de « portail Etudiant » permettant de valider les modèles et les architectures étudiés. Ce portail étudiant constituera un système d'informations personnalisées qui intégrera dans un premier temps deux types de services :
  - l'accès aux informations de base « universitaires » et des ressources pédagogiques numériques
  - l'accès à des services connexes mais aussi sur la mobilité, dans un but d'accompagnement de la population estudiantine dans leurs « univers » et leurs activités
- Informer ESPERE du choix de l'utilisateur pour que PerEspere mette à jour les connaissances qu'il a de l'étudiant.

5.1.1.2. Structuration des packages

Les différentes fonctionnalités du système peuvent être regroupées en trois packages :

- **Modules :** ce package regroupe toutes les fonctionnalités qui participent dans la recherche des modules validés par l'étudiant et des modules disponibles qui peuvent le concernés.
- **Recherche de stages :** ce package regroupe toutes les fonctionnalités qui participent dans la recherche personnalisée et non personnalisée de stages et des fonctionnalités qui informe automatiquement l'étudiant des stages qui peuvent l'intéressés.
- **Emplois et rattrapage:** ce package regroupe toutes les fonctionnalités qui informe l'étudiant de ces emplois de temps, les modifications et les éventuels rattrapages.
- **Recherche bibliothécaire :** ce package regroupe toutes les fonctionnalités qui participent dans la recherche de personnalisés de documents et la consultation de la bibliothèque.
- **Recherche d'itinéraires :** ce package regroupe toutes les fonctionnalités qui participent dans la recherche personnalisée et non personnalisée d'itinéraires.
- **XML/BD :** ce package regroupe toutes les fonctionnalités qui permet la gestion de toutes les bases de données de notre système, ainsi que le passage de l'XML aux base de données et vise vers ça lors de toutes communication entre ESPERE et PerEspere.
- **Service web "PersonnalizeSystem"** : ce package s'occupe de toutes les fonctionnalités en relation avec la communication entre ESPERE et PerEspere à travers le service web « PersonnalizeSystem ».
- **Administration et configuration du système ESPERE** : regroupe les actions concernant l'authentification de l'étudiant pour assurer l'apprentissage

5.1.2. Les cas d'utilisations

Les fonctionnalités du système sont décrites à partir des diagrammes des use-cases du langage de modélisation objet UML.

Dans ce qui suit, on présente au premier lieu le diagramme complet des ces cas d'utilisation.Nous détaillons, par la suite, le package « Module ». Le reste de la spécification est attacher en annexe e ce rapport.

Le diagramme des cas d'utilisations complet récapitule les principales fonctionnalités du système et les principaux acteurs qui interagis avec ce système. Notamment le système de personnalisation multi-agent (PerESPERE) est considéré comme un acteur puisqu'il interagie avec le système ESPERE et appartient à l'environnement extérieur de ce dernier.

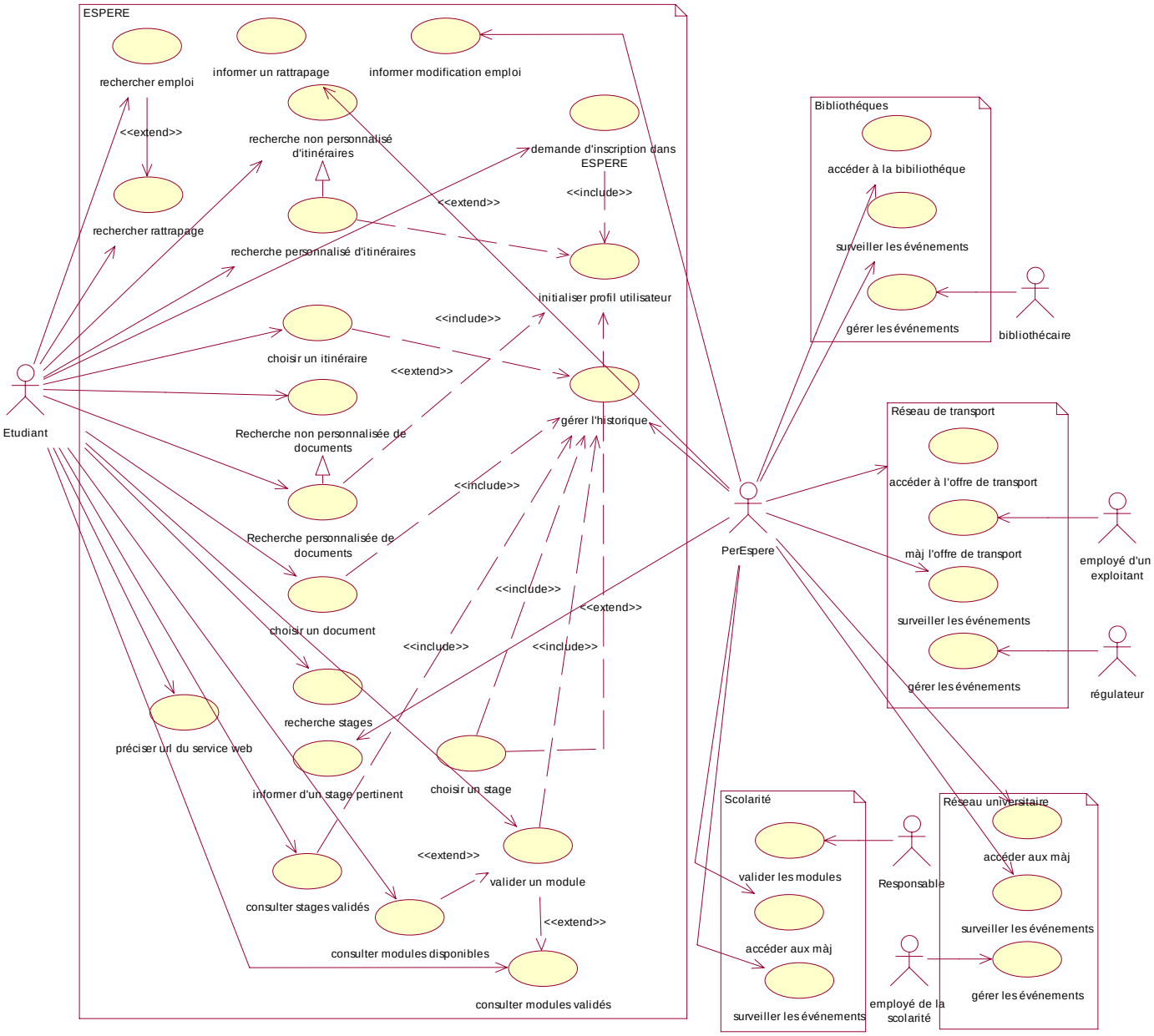


Figure 5.1. Diagramme des cas d'utilisation complet du système ESPERE.

5.1.2.1. Package « Stages »

Cas d'utilisations du package « Stages »

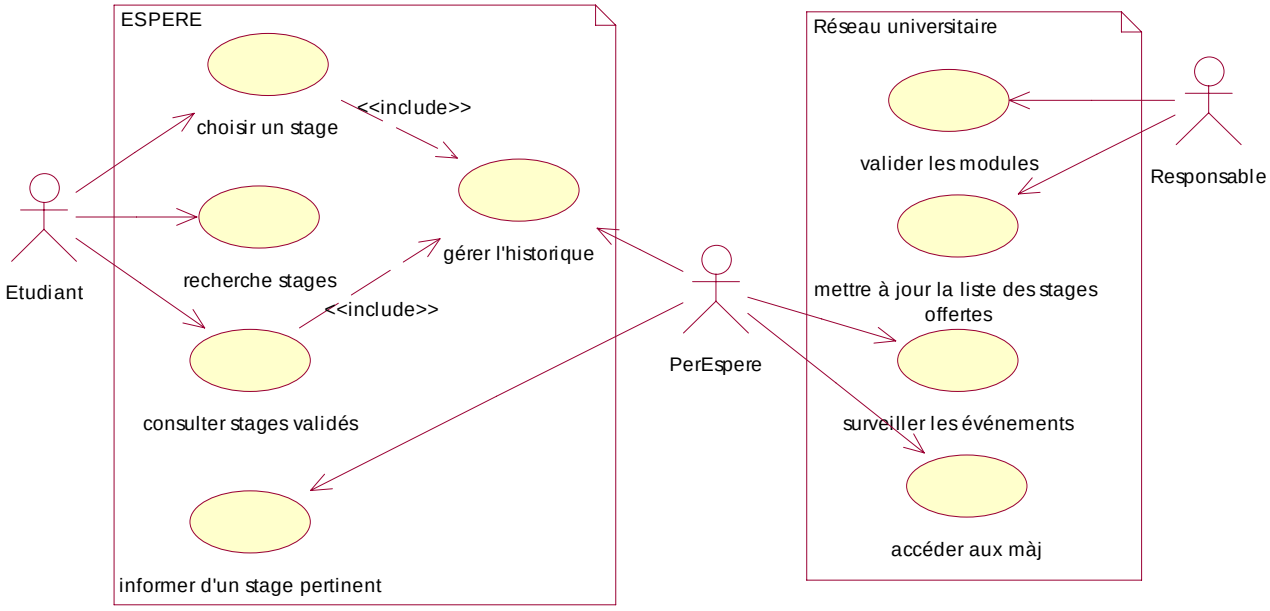


Figure 5.2. Diagramme des cas d'utilisation du package Stages.

Le package « Stages » récapitule toutes les fonctionnalités qui participent dans la recherche personnalisée et non personnalisée de stages et des fonctionnalités qui informe automatiquement l'étudiant des stages qui peuvent l'intéressés.

Description des principaux cas d'utilisation

❖ *Cas d'utilisation << Informer d'un stage pertinent >>*

Use case *« Informer d'un stage pertinent »*
Pré-conditions : ▪ PerEspere est en recherche permanente des éventuels stages proposés.
▪ PerEspere détecte un nouveau stage. ▪ PerEspere cherche la liste de tous les étudiants qui peuvent être concernés par ce stage. ▪ PerEspere diffuse une requête vers tous les étudiants concernés. ▪ PerEspere reprend la recherche permanente de stages
Exception *:*

❖ Cas d'utilisation <<choisir un stage>>

Use case *« Choisir un stage »*
Pré-conditions : ▪ L'étudiant est déjà inscrit au système ESPERE. ▪ Fichier de configuration contenant l'URL du service web disponible.
▪ L'étudiant consulte le système ESPERE. ▪ Demande la liste des stages qui peuvent le concernés. ▪ ESPERE appel le service web. ▪ Le service web transmet la requête vers PerEspere. ▪ PerEspere cherche toutes les solutions possibles. ▪ PerEspere sélecte les stages pertinents pour l'étudiant. ▪ PerEspere envoie la réponse vers le service web. ▪ Le service web envoie la réponse vers ESPERE. ▪ L'étudiant consulte la liste des solutions possibles. ▪ L'étudiant choisie le stage le plus pertinent. ▪ ESPERE envoie une requête vers PerEspere contenant le choix de l'étudiant ▪ PerEspere mais à jour l'historique de l'étudiant
Exception *:* ▪ Le service web est introuvable à l'adresse précisée.
Extension *:* ▪ L'étudiant consulte l'ensemble des stages qui a déjà validés. ▪ L'étudiant demande la consultation de tous les stages disponibles.
Post-conditions :

5.2. Analyse et conception

5.2.1. Dépendance entre les packages

Pour la clarté du diagramme des classes d'entités, il est nécessaire de montrer au premier lieu la dépendance entre les différents packages puis présenter les diagrammes de classes de chaque package indépendamment dans la phase de la conception.

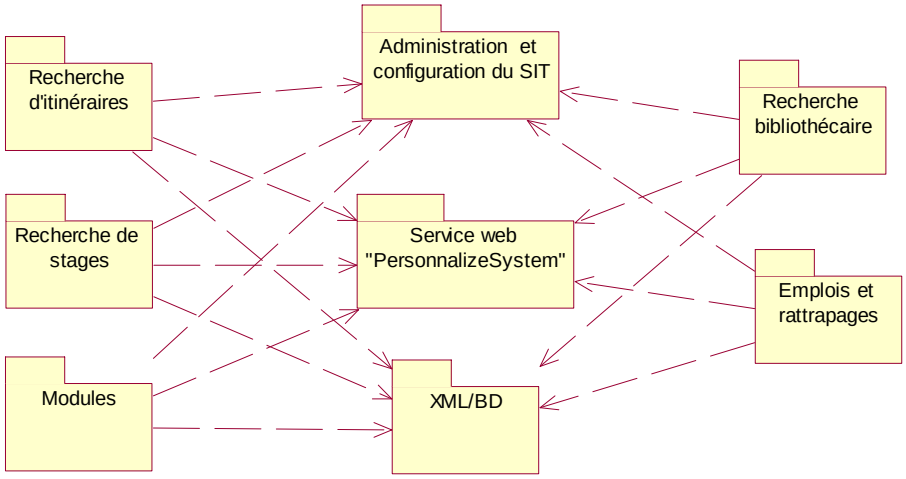


Figure 5.3. Diagramme de dépendance entre les packages.

Nous présentons dans cette section, la conception du package « Stages ». La suite de la conception et les détailles de tous les packages de notre système sont attachés en annexes de ce rapport.

5.2.2. Package « Recherche de stages»

5.2.2.1. Diagramme des classes

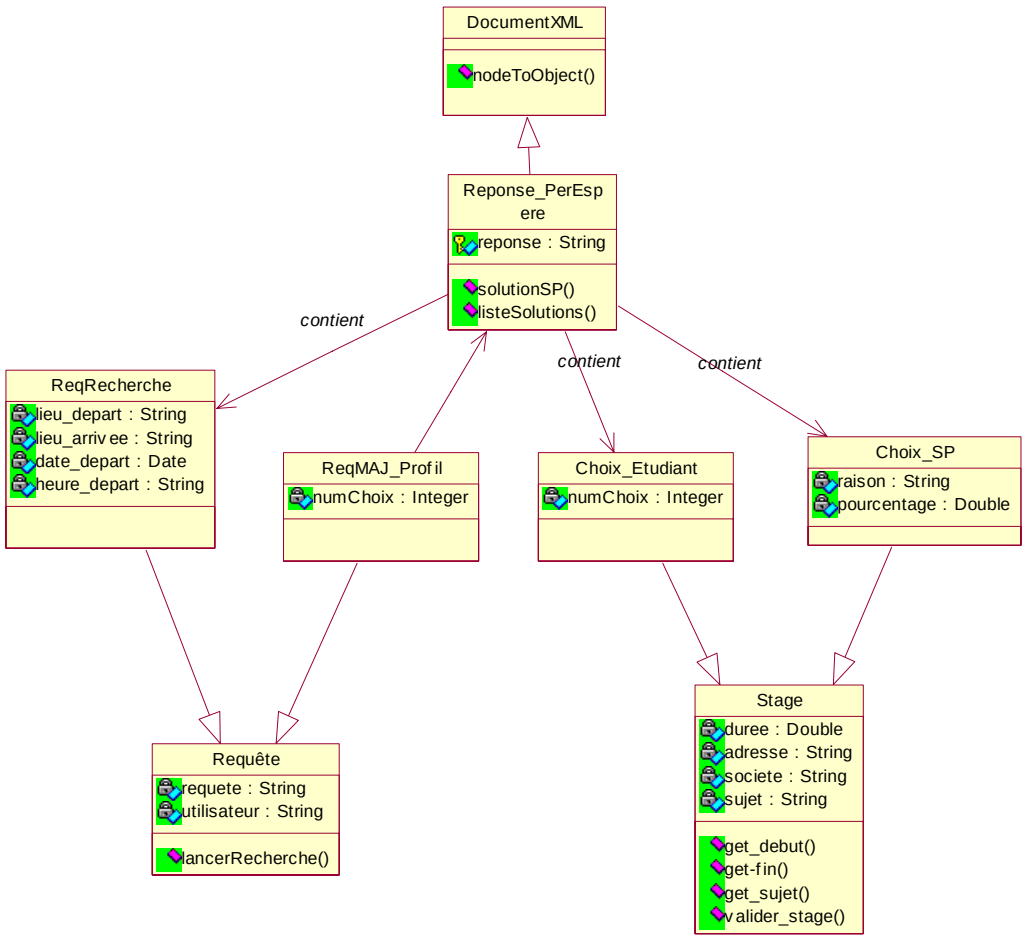


Figure 5.4. Diagramme de classe du package Recherche des stages.

5.2.2.2. Description des méthodes principales

- **lancerRecherche()** : méthode invoquée par l'étudiant pour lancer une requête de recherche de stage.
- **solutionSP()** : méthode invoquée lors de la réception d'une réponse à une requête de recherche pour extraire le stage préféré proposée par SP.
- **listeChoix()** : méthode invoquée pour extraire à partir de la réponse du SP la liste de tous les itinéraires possibles entre le pont de départ et le point d'arrivée préciser par l'utilisateur.

5.2.2.3. Diagrammes de séquences

❖ *Cas d'utilisation << Recherche personnalisée de stage >>*

Ce diagramme de séquence permet la recherche de stage qui répond aux préférences de l'étudiant en se basant sur les historiques.

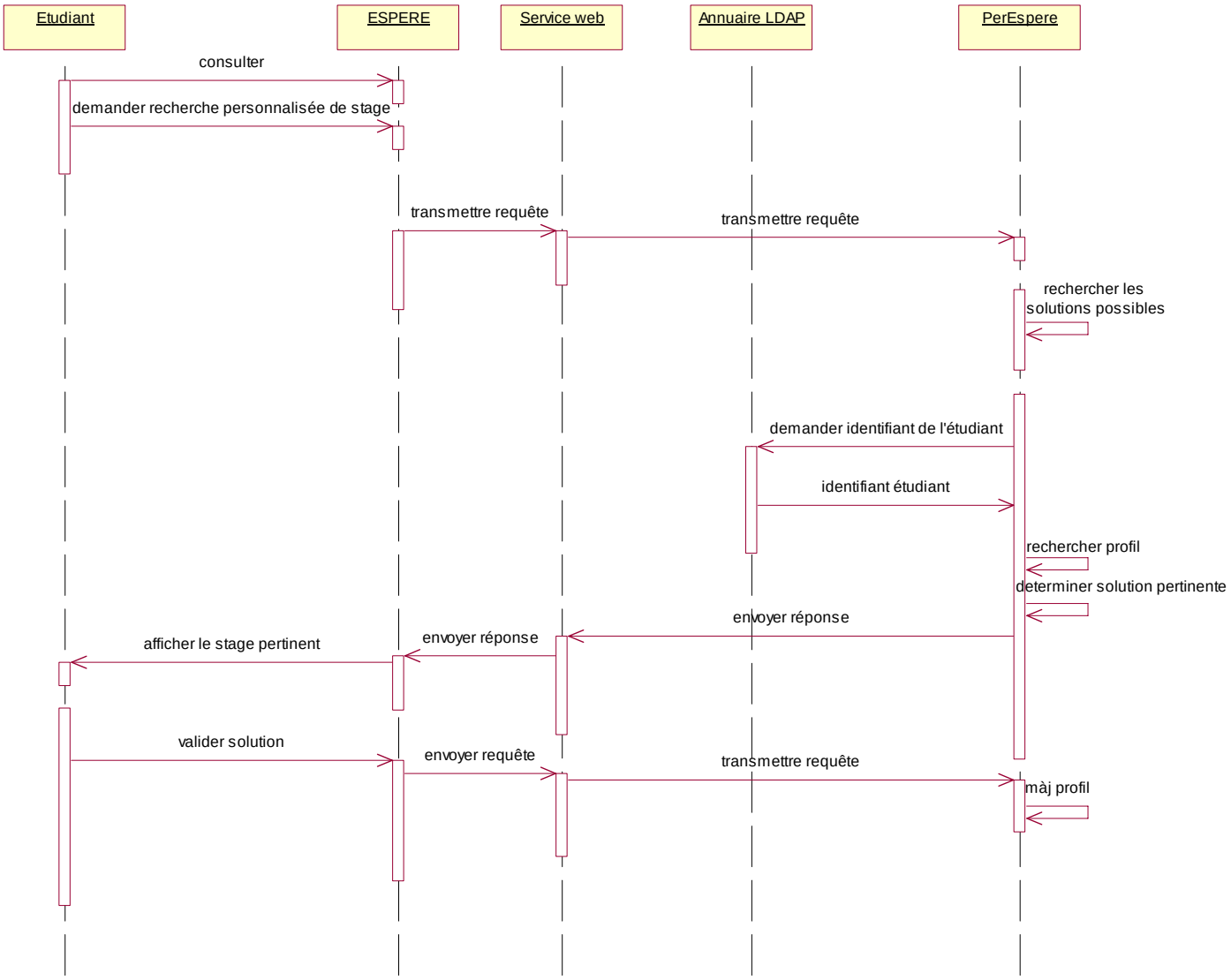


Figure 5.5. Diagramme de séquence « Recherche personnalisée de stage ».

❖ *Cas d'utilisation << Recherche non personnalisée de stages >>*

Ce diagramme de séquence donne à l'utilisateur tous les stages disponibles en précisant la période, le sujet et l'endroit.

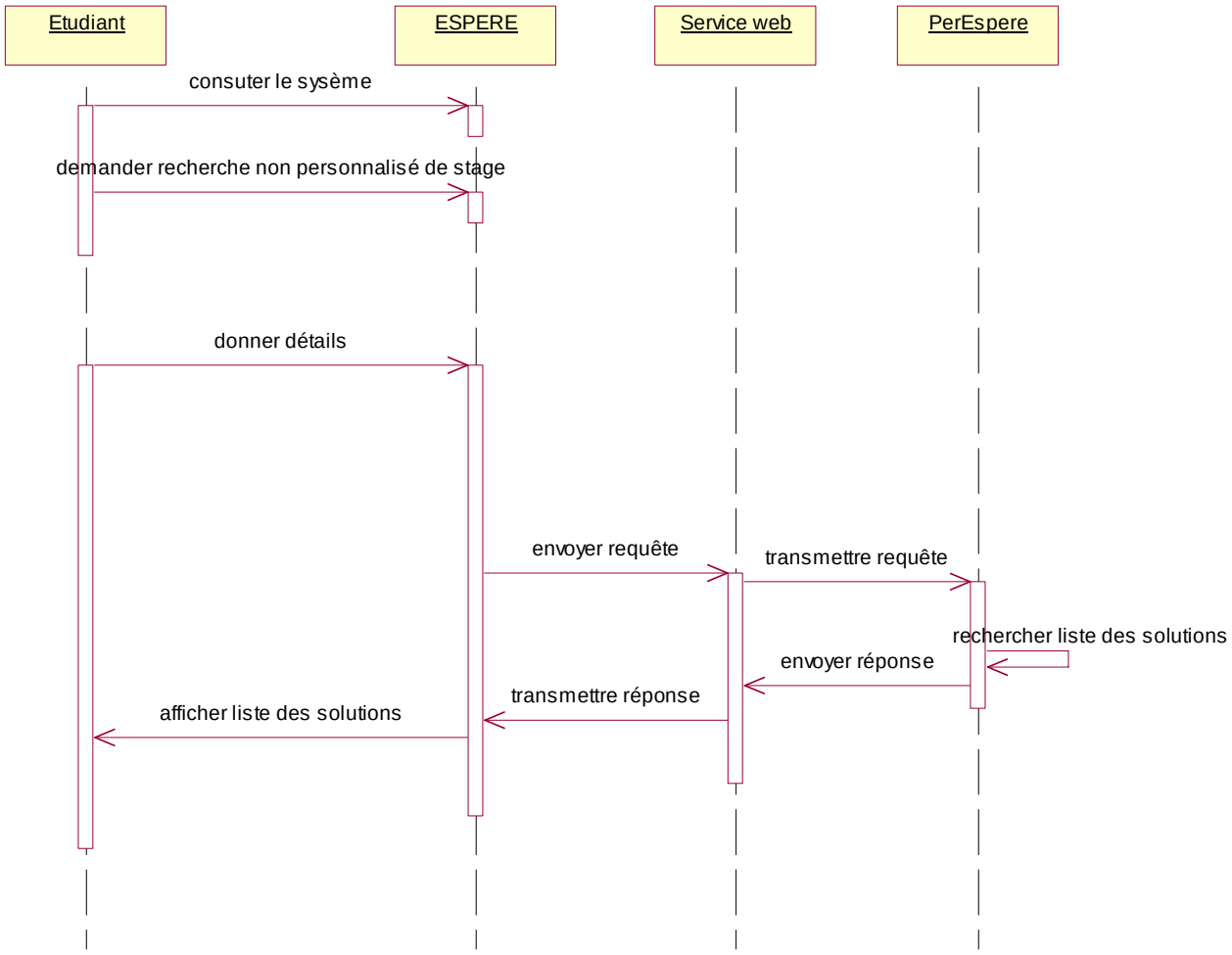


Figure 5.6. Diagramme de séquence « recherche non personnalisée de stage ».

❖ *Cas d'utilisation << Choisir un stage >>*

S'il effectue une recherche de stage personnalisée, l'étudiant peut soit accepter la solution offerte par le système de personnalisation, soit sélectionner une autre solution qui le plait encore plus. Et c'est grâce à ce diagramme de séquence que le PerESPERE prend en compte le choix de l'étudiant pour la mise à jour de son profil.

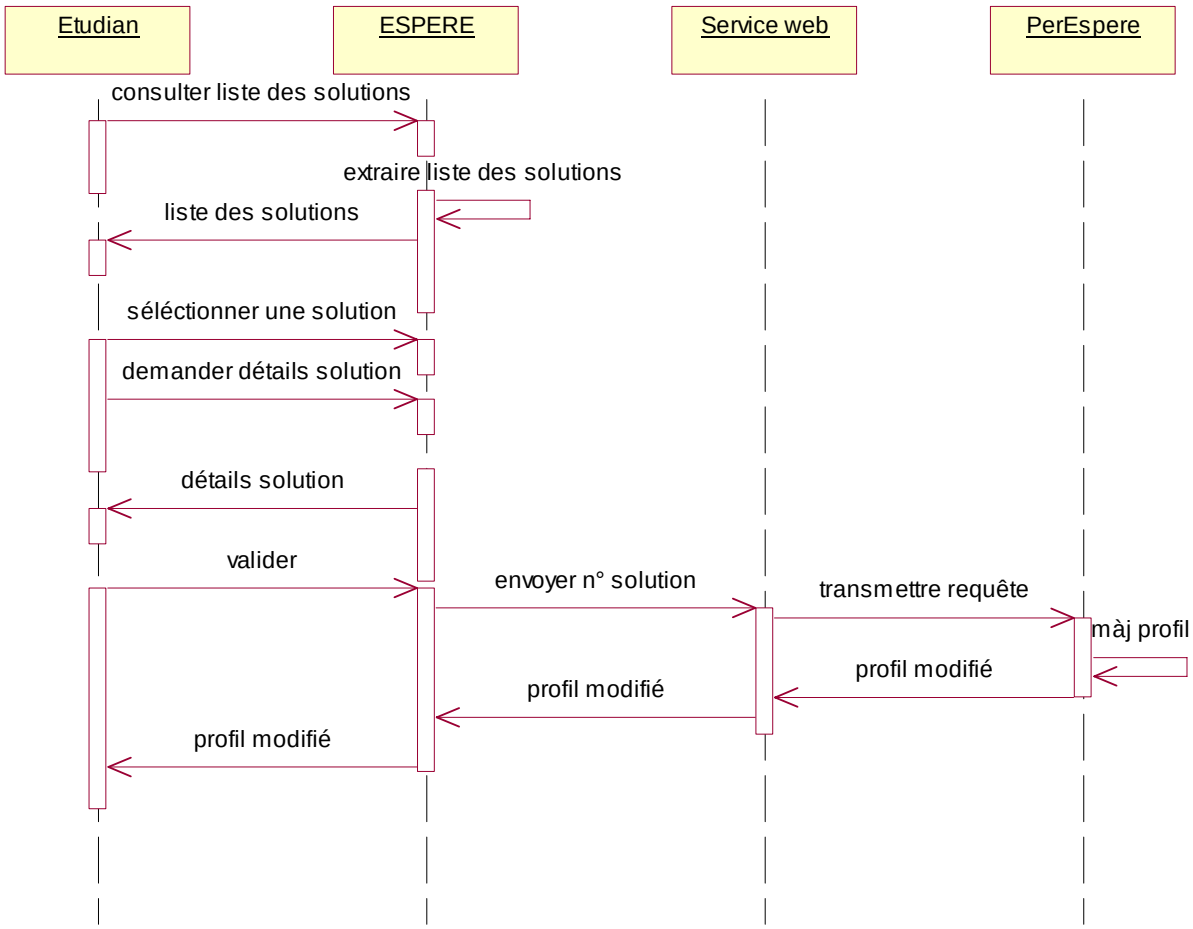


Figure 5.7. Diagramme de séquence « choisir un stage ».

5.3. Réalisation pratique

Dans cette partie, on va présenter quelques séquences d'exécution de notre système ESPERE avant et après l'intégration avec notre système PerESPERE pour pouvoir détecter les apports de la personnalisation de l'information apporter par le système de personnalisation. Pour illustrer convenablement notre système, on va présenter quelques interfaces tout en détaillant le processus des tâches importantes réalisées par ESPERE.

Les détails sur l'environnement de travail appliqués et les différentes plateformes utilisées sont précisés en annexes.

5.3.1. Inscription d'un étudiant

Lorsque l'étudiant fait appel au système ESPERE pour s'inscrire deux types d'inscription sont faites en parallèle :

- L'inscription dans la base de données « universitaire » : l'identifiant unique de l'étudiant dans cette base de données est le numéro de CIN (carte d'identité nationale).
- L'inscription dans l'annuaire LDAP du système de personnalisation PerEspere : cette inscription se fait automatiquement lors de l'inscription de l'étudiant dans ESPERE. Et c'est grâce à cette authentification que PerEspere peut par la suite personnaliser les recherches du système ESPERE, comme il peut informer automatiquement l'étudiant de toutes les nouveautés et sujets qui l'intéressent.

Lors de la première visite, ESPERE demande de l'étudiant de s'inscrire au système. La figure 5.6 représente l'interface d'inscription, cette interface contient trois zones essentielles:

- Zone 1 : qui est-il/ elle ? il s'agit ici de cerner le profil de l'utilisateur sur la base de quelques données personnelles (âge, adresses, handicap, abonnement de transports collectifs, etc). Tous les champs de cette zone sont obligatoirement accomplis par l'étudiant.
- Zone 2 : typiquement, les lieux fréquentés (domicile études, domicile parents, lieu d'études,…), comprenant également les lieux/moyens d'accès à l'information universitaire.
- Zone 3 : cette zone joue deux rôles :
  - Avant de s'inscrire au système : ESPERE informe automatiquement les visiteurs du système (les étudiants) de toutes les nouveautés universitaires (stages proposés, nouveaux documents, perturbation dans le réseau de transport, les inscriptions universitaire,…). Dans ce ces, on a pas des notions de personnalisations de l'information puisque l'utilisateur n'est pas encore inscrit dans le système.
  - Après l'inscription : ESPERE continu à informer automatiquement l'étudiant des nouveautés, mais sauf les nouveautés qui peuvent le concernées, en d'autres termes, informer l'étudiant des informations et sauf les informations pertinentes aux choix et orientation de l'étudiant.

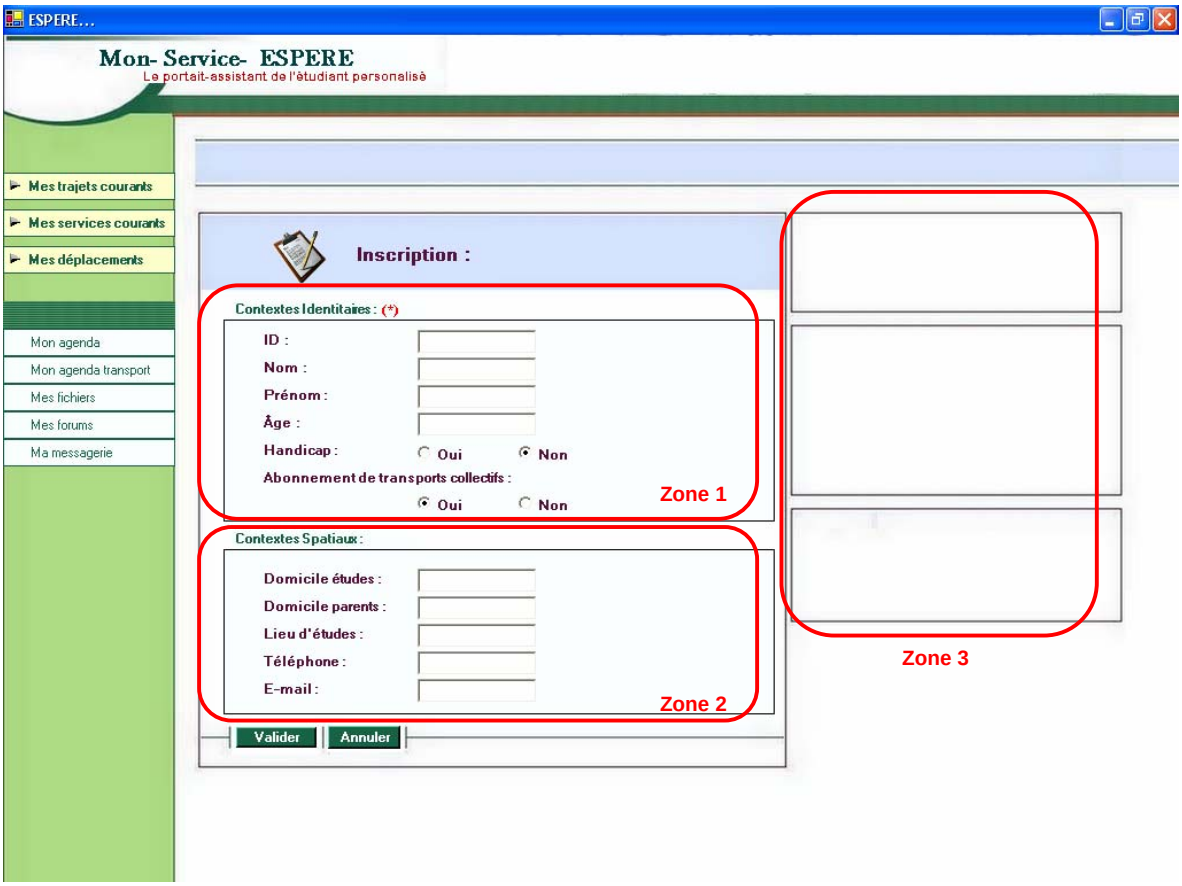


Figure 7.8. Inscription d'un étudiant.

5.3.2. Les services courants

Notre système a été réalisé de façon incrémentale afin d'intégrer progressivement des fonctionnalités de niveaux de plus en plus élevés, pour se fait, on a réalisé les quatre principales services de notre système qui peuvent être incrémenter selon les besoins de l'application.

En effet, notre système implique les services courants suivant :

- Gestion des modules
- Emplois et rattrapages
- Recherche bibliothécaire
- Recherche d'itinéraires

Ces quatre services seront par la suite détaillés.

5.3.3. Gestion des modules

Pour gérer ces modules, un étudiant peut utiliser les fonctionnalités offertes par le système de personnalisation (utiliser les informations personnalisées), comme il peut prendre ses propres décisions sans faire appel au système de personnalisation. En effet, ESPERE offre trois fonctionnalités essentielles pour la gestion des modules :

- **Consulter mes modules validés** : il s'agit d'une simple consultation des modules déjà validés par l'étudiant. En plus c'est grâce aux modules déjà validés (historiques) que PerEspere filtre les informations pour personnaliser la gestion des modules.
- **Consulter liste des modules** : cette fonctionnalité permet à l'étudiant de visualiser toutes les modules est choisir ceux qui lui convient sans intervention du système.
- **Proposer un module** : cette fonctionnalité se base essentiellement sur le système de personnalisation PerEspere. En fait, PerEspere consulte les modules de l'utilisateur qui a déjà validés, comme il consulte le profil de l'étudiant qui informe le système des préférences et goûts de l'étudiant. Par la suite, PerEspere filtre la liste des modules pour extraire les modules qui intéresse le plus l'étudiant concerné.

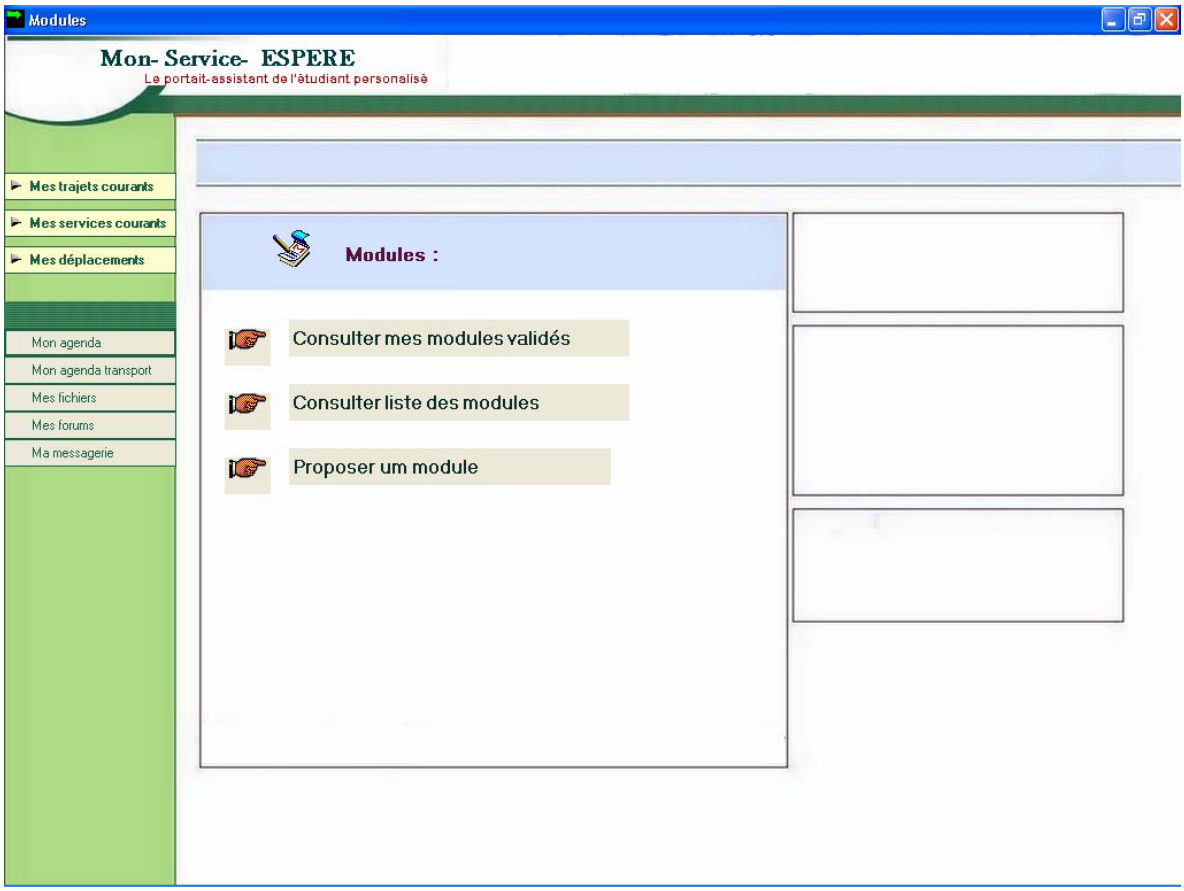


Figure 5.9. Gestion des modules.

5.3.4. Emplois et rattrapages

Lors de chaque modification dans les emplois de temps ou d'un rattrapage, le système de personnalisation cherche la liste de tous les étudiant concernés par ces modifications pour les informer automatiquement.

La sélection des utilisateurs ne se base pas simplement sur les étudiant possèdent nécessairement cet emploi de temps, mais aussi elle incluse ceux motivés par la thématique ou le sujet de la matière concerné par la modification.

En outre, ESPERE donne la main à l'utilisateur de faire ses propres recherches, pour trouver toutes les modifications effectuées sur son emploi, comme il peut chercher les rattrapages par date et/ou par matière.

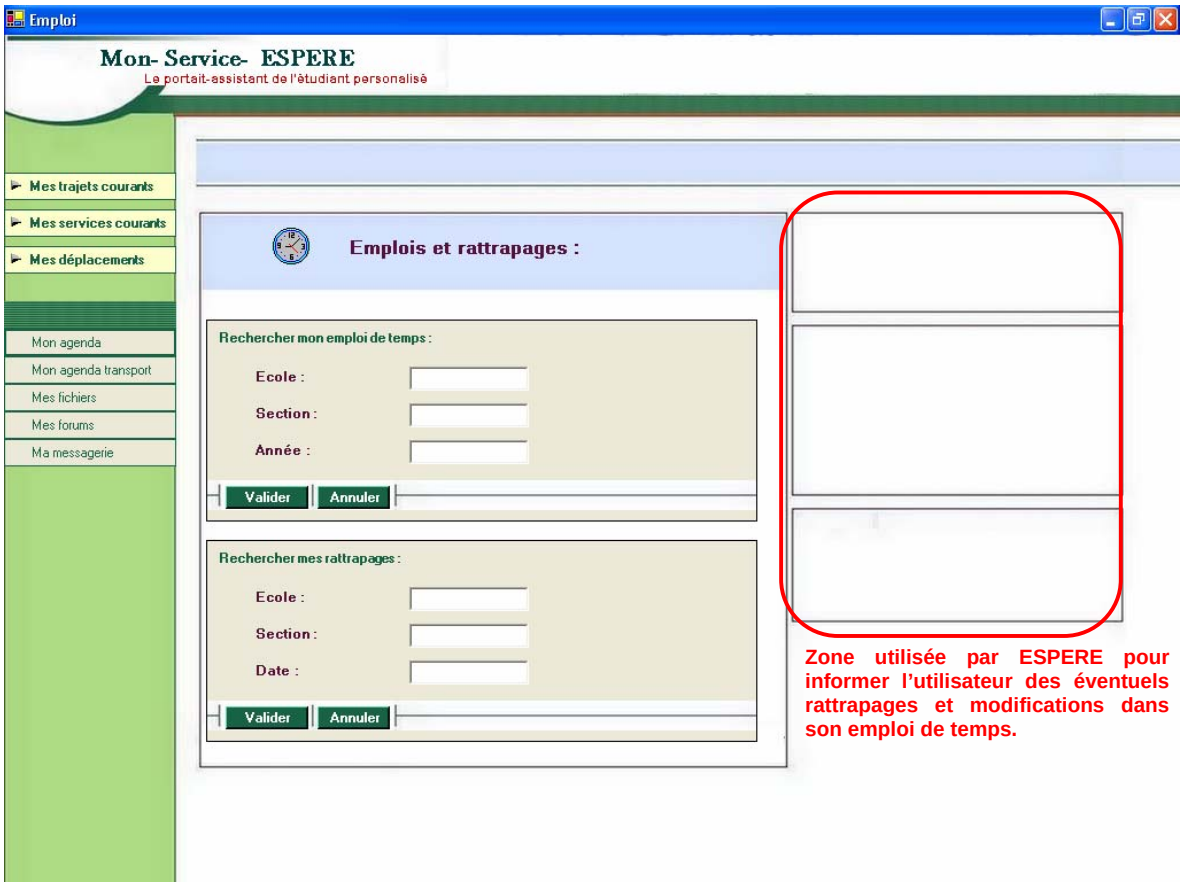


Figure 5.10. Emplois et rattrapage.

5.3.5. Recherche bibliothécaire

5.3.6. Recherche d'itinéraires

Pour trouver le trajet pertinent aux contraintes, choix et préférences de l'étudiant, il suffit de remplir les champs de la fenêtre qui s'affiche en choisissant le service déplacement (figure 5.11), ESPERE récupère alors l'identifiant de l'étudiant formule la requête de l'utilisateur selon un

format XML (voir figure 5.10) compatible avec PerESPERE, l'envoie de la requête se fait par un appel synchrone du service web.

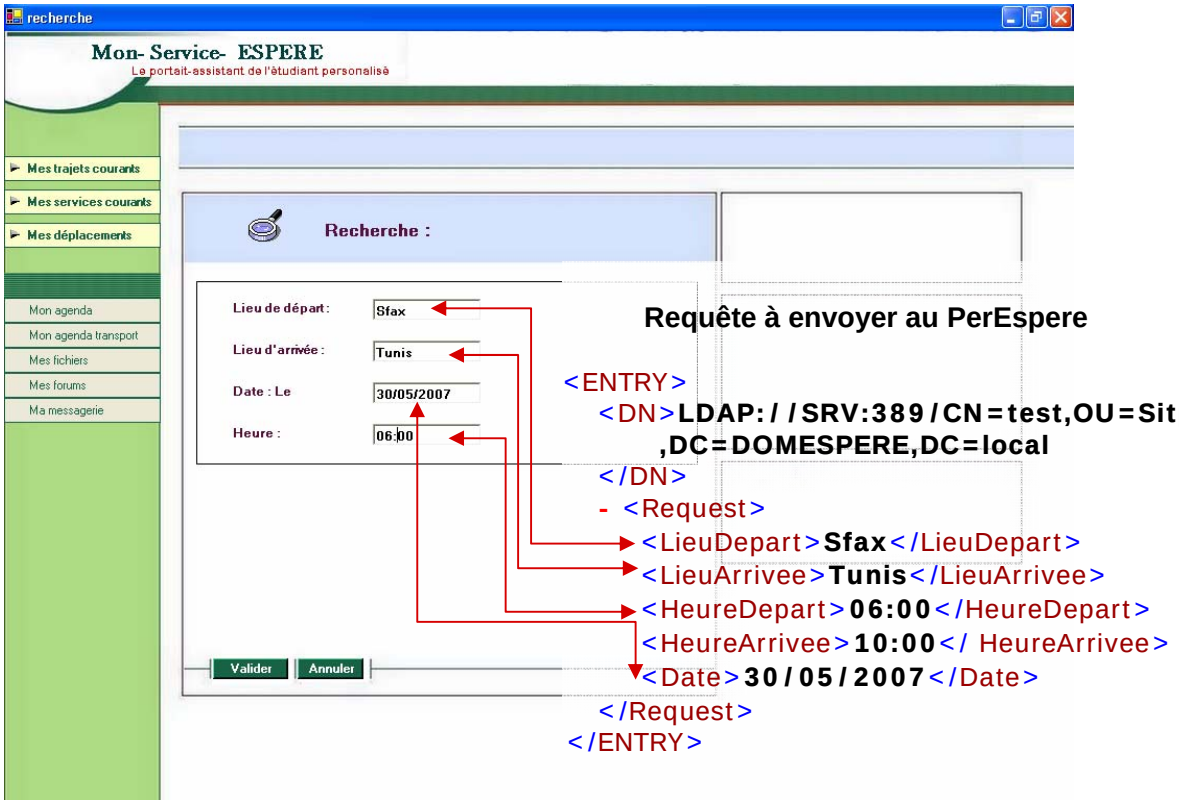


Figure 5.11. Recherche d'itinéraires.

Conclusion

La réalisation du système ESPERE a été effectuée pour mettre en œuvre et évaluer des services liés à la personnalisation des informations. Ces services seraient centrés sur l'usage des services et la mobilité au quotidien, dans un but d'accompagnement du jeune étudiant dans son « univers » et ses activités. Plus précisément, dans ce chapitre nous avons développer un démonstrateur capable de :

- Extraire des gisements de données de l'information pertinente pour les étudiants
- Présenter l'information de manière immédiate et naturelle. Toutes en distinguant l'information statique et l'information temps réel,

Les objectifs qui restent à réaliser dans l'ordre de les intégrer à notre système « ESPERE » qui est un système évolutif et incrémental sont :

- Diffuser l'information sur plusieurs supports, tout en assurant une présentation ergonomique et un contenu homogène.
- Organiser la mobilité quotidienne en fonction de l'agenda de l'étudiant fin de gagner du temps et mieux organiser ses activités et son déplacement,

Conclusion générale

Conclusion générale

Dans ce mémoire, nous avons étudié les systèmes d'information personnalisée (SIP). Nous avons, dans un premier temps, étudié l'apport de la personnalisation pour les systèmes d'information. Cette étude a mené à définir la personnalisation comme l'adaptation des données aux besoins, aux préférences, aux capacités des utilisateurs et aux contextes des requêtes. La personnalisation des informations en tant qu'adaptation à l'utilisateur ainsi que la prise en compte de l'aspect distribué des informations nous ont mené aux agents logiciels.

Dans le second chapitre, nous avons présenté trois approches de personnalisation des systèmes d'informations. Les deux premières concernent la reformulation de la requête de l'utilisateur dans un système multi-sources et la troisième permet la personnalisation des systèmes à données homogènes.

Par la suite, nous avons proposé une approche de personnalisation de l'information qui possède les avantages des ces approches : reformulation de la requête et séparation du SI et SP. Nous avons basé, pour cela, sur l'approche de médiation personnalisée où nous avons proposé une conception d'un médiateur pour l'intégration des systèmes à sources de données hétérogènes et distribuées.

L'étude et la réalisation du système de personnalisation « PerESPERE » à base d'agent logiciel ont fait l'objectif du chapitre 4. Nous avons alors présenté la plateforme multi-agents adéquates et l'architecture générale du système. Tandis que, la conception du médiateur « MedESPERE » a fait l'objectif du cinquième chapitre.

Ces deux systèmes ont été par la suite utilisé pour personnaliser notre Système ESPERE : un portail qui englobe les essentiels fonctionnalités qui aident l'étudiant tout au long de son cycle universitaire. ESPERE est un démonstrateur itératif et incrémental qu'on peut lui greffer toutes les fonctionnalités qu'on a besoin sous forme de « service ».

À partir de la maquette réalisée dans ce cadre, nous avons effectué les premières évaluations de notre système et mis en évidence les évaluations techniques et ergonomiques restant à faire.

Après la validation complète de nos travaux, nous envisageons d'étendre le système ESPERE pour être indépendant de la plateforme d'accueil. Ainsi, on a besoin de se débarrasser de l'interface figée, pour parler de interfaces adaptatives et avoir ainsi, un système d'information personnalisé de point de vue contenu et contenant.

Des extensions sur le système de médiation « MedESPERE » pourraient être réalisées dans le contexte d'amélioration du temps de réponse de ce système et de l'optimisation de sous requêtes générées. En fait, l'optimisation de la requête porte sur différents niveaux : utilisation des règles d'équivalence pour réécrire les requêtes, génération de plans d'exécution optimisés, etc.

Annexes

I. Quelques notions

Dans cette section, nous présentons quelques notions qui utilisées pour assurer la communication entre SI et SP d'une part, et permettant d'identifier les utilisateurs d'une autre.

I.1. Les Services Web

I.1.1 Qu'est ce qu'un Web Service

Un Service Web est un composant implémenté dans n'importe quel langage, déployé sur n'importe quelle plate-forme et enveloppé dans une couche de standards dérivés du XML (eXtended Markup Language). Il doit pouvoir être recherché et invoqué dynamiquement par d'autres services.

L'XML est à base de tous les protocoles décrits ci-dessous. Le fait que les Web Services utilisent l'XML leurs procurent l'avantage d'être non propriétaire et ainsi réellement multi plateforme.

Le concept des Web Services s'articule actuellement autour des deux acronymes suivants :

- SOAP (Simple Object Access Protocol) est un protocole d'échange inter applications indépendant de toute plate-forme, basé sur le langage XML. Un appel de service SOAP est un flux ASKII encadré dans des balises XML et transporté dans le protocole http.
- WSDL (Web Services Description Langage) donne la description au format XML des Web Services en précisant les méthodes pouvant être invoquées, leurs signatures et le point d'accès (URL, port, etc…). C'est, en quelque sorte, l'équivalent du langage IDL pour la programmation distribuée CORBA.

I.1.2 Le protocole SOAP

SOAP est un protocole à la fois simple et léger destiné à l'échange d'informations. SOAP diffère de RMI, CORBA et COM car il concentre les informations et utilise le principe d'auto description des données. SOAP fait partie de la couche de communication des Web Services. La force de ce protocole réside dans son université et sa flexibilité. Il définit la structure des messages XML utilisés par les applications pour dialoguer entre elles. SOAP fait figure de pièce centrale parmi tous les protocoles évoqués.

La communication avec les Web Services s'effectue via le protocole SOAP. SOAP peut être utilisé soit pour l'appel de méthodes (SOAP RPC), doit pour l'échange de message (SOAP Messaging).

SOAP définit la structure principale du message, dite « enveloppe » qui est composé de deux parties : l'entête qui est facultative et le corps qui est obligatoire (figure 1).

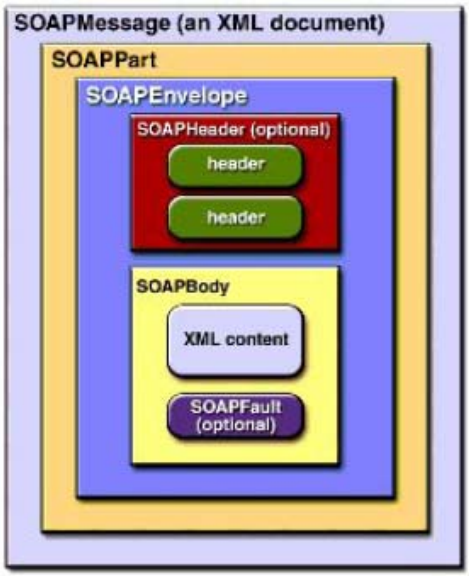


Figure 1. Structure principale d'un message SOAP.

I.1.3 WSDL : Données et Sécurité

C'est un format de description des méthodes et des paramètres des composant invocables par le biais des messages au format SOAP. WSDL est une grammaire dérivée de l'XML. L'échange de données s'appuie sur le protocole SOAP pour l'échange des messages et sur le langage WSDL pour la définition du contrat de l'interface. WSDL comprend la définition de plusieurs données : type, message, opertion , portType, port, service et binding.

WSDL est un système de communication « point à point ». Un consommateur du Service Web interroge le serveur, sur lequel celui-ci est disponible, et celui-ci retourne la description du Service Web demandé.

I.2. Le protocole LDAP

LDAP (Lightweight Directory Access Protocol, Protocole d'accès aux annuaires léger) est un protocole standard permettant de gérer des annuaires, c'est-à-dire d'accéder à des bases d'informations sur les utilisateurs d'un réseau par l'intermédiaire de protocoles TCP/IP.

Le protocole LDAP définit la méthode d'accès aux données sur le serveur au niveau du client, et non la manière de laquelle les informations sont stockées.

LDAP présente les informations sous forme d'une arborescence d'informations hiérarchique appelée DIT (Directory Information Tree), dans laquelle les informations, appelées entrées (ou encore DSE, Directory Service Entry), sont représentées sous forme de branches. Une branche située à la racine d'une ramification est appelée racine ou suffixe.

Chaque entrée de l'annuaire LDAP correspond à un objet abstrait ou réel, constitué d'un ensemble de paires clés/valeurs appelées attributs.

II. Spécification

II.1. Package « Emplois et rattrapages »

- ***Cas d'utilisations du package « Emplois et rattrapage »***

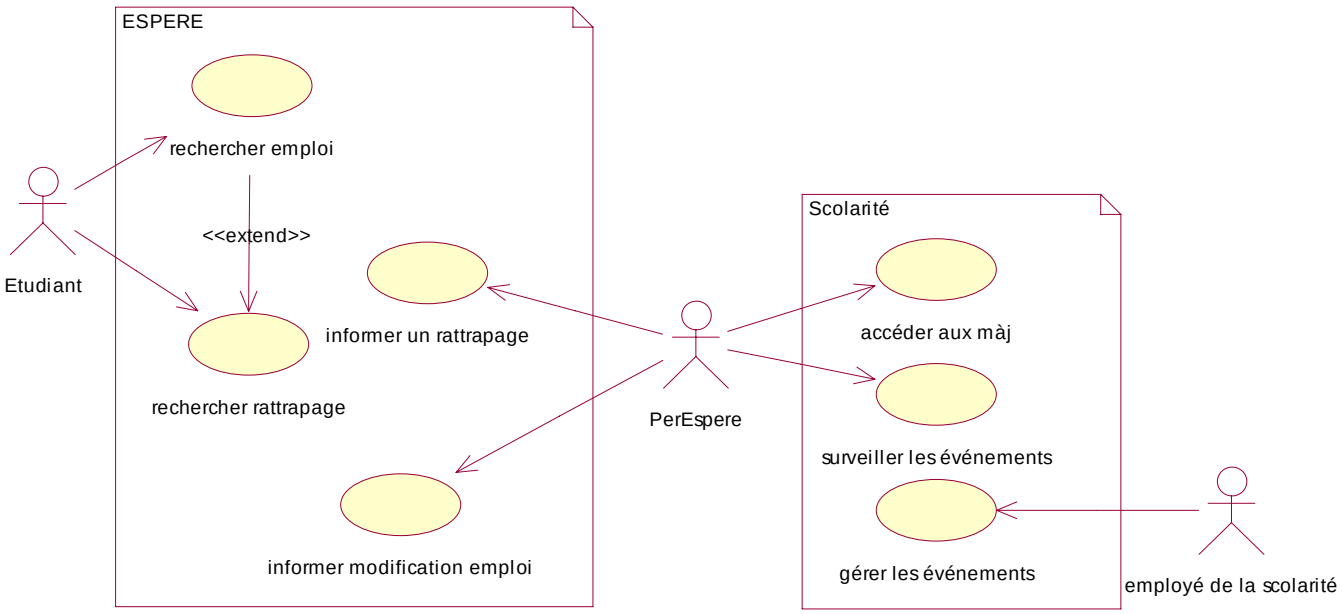


Figure 2. Diagramme des cas d'utilisation du package Emplois et rattrapages.

Le package « Emplois et rattrapage » récapitule toutes les fonctionnalités qui informe l'étudiant de ces emplois de temps, les modifications et les éventuels rattrapages.

- ***Description des principaux cas d'utilisation***

❖ *Cas d'utilisation << Informer d'un rattrapage >>*

Use case *« Informer d'un rattrapage »*
Pré-conditions : ▪ PerEspere est en recherche permanente des éventuels rattrapages.
▪ PerEspere détecte un nouveau rattrapage. ▪ PerEspere cherche la liste de tous les étudiants concernés par ce rattrapage. ▪ PerEspere diffuse une requête vers tous les étudiants concernés. ▪ PerEspere reprend la recherche permanente de rattrapages
Exception *:*

❖ *Cas d'utilisation << Rechercher emploi >>*

Use case *« rechercher emploi »*
Pré-conditions : ▪ L'étudiant est déjà inscrit au système ESPERE. ▪ Fichier de configuration contenant l'URL du service web disponible.
▪ L'étudiant demande la recherche de son emploi de temps. ▪ ESPERE appel le service web. ▪ Le service web transmet la requête vers PerEspere. ▪ PerEspere cherche l'emploi de temps demandé. ▪ PerEspere envoie la réponse vers le service web. ▪ Le service web envoie la réponse vers ESPERE. ▪ L'étudiant consulte son emploi de temps.
Exception *:* ▪ Le service web est introuvable à l'adresse précisée.
Extension *:* ▪ L'étudiant demande la recherche des éventuels rattrapages

II.2. Package « Modules »

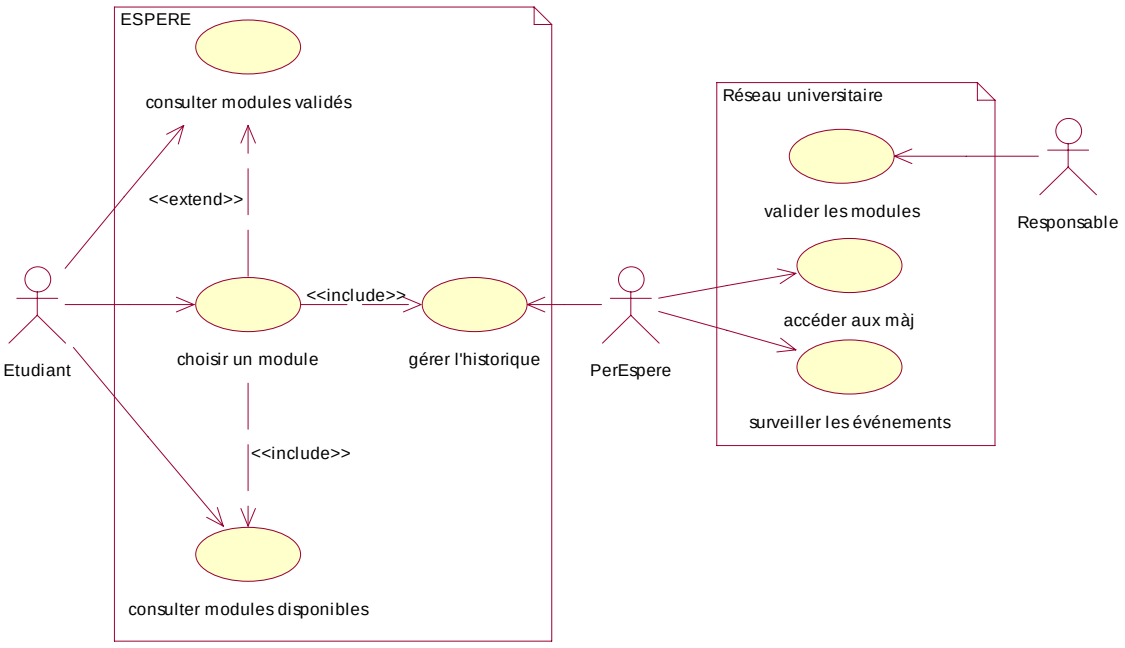


Figure 3. Diagramme des cas d'utilisation du package Modules.

Le package « Modules » récapitule toutes les fonctionnalités qui participent dans la recherche personnalisée et non personnalisée de stages et des fonctionnalités qui informe automatiquement l'étudiant des stages qui peuvent l'intéressés.

Description des principaux cas d'utilisation

- *Cas d'utilisation << Choisir un module >>*

Use case *« Choisir un module »*
Pré-conditions : ▪ L'étudiant est déjà inscrit au système ESPERE. ▪ Fichier de configuration contenant l'URL du service web disponible.
▪ L'étudiant consulte le système ESPERE. ▪ Demande la liste des modules qui peuvent le concernés. ▪ ESPERE appel le service web. ▪ Le service web transmet la requête vers PerEspere. ▪ PerEspere cherche toutes les solutions possibles. ▪ PerEspere sélecte les modules pertinents pour l'étudiant. ▪ PerEspere envoie la réponse vers le service web. ▪ Le service web envoie la réponse vers ESPERE. ▪ L'étudiant consulte la liste des solutions possibles. ▪ L'étudiant choisie le module le plus pertinent. ▪ ESPERE envoie une requête vers PerEspere contenant le choix de l'étudiant ▪ PerEspere mais à jour l'historique de l'étudiant
Exception *:* ▪ Le service web est introuvable à l'adresse précisée.
Extension *:* ▪ L'étudiant consulte l'ensemble des modules qui a déjà validés. ▪ L'étudiant demande la consultation de tous les modules disponibles.
Post-conditions :

II.3. Package « Service web "PersonnalizeSystem"»

— ***Cas d'utilisations du package « Service web "PersonnalizeSystem"»***

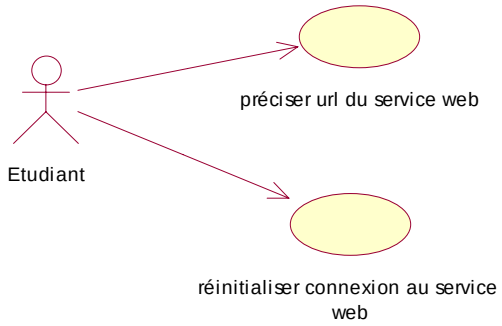


Figure 4. Diagramme des cas d'utilisation du package Service web « PersonnalizeSystem ».

Le package « Service web PersonnalizeSystem » récapitule les fonctionnalités qui correspond à gestion de la communication via le service web entre ESPERE et PerEspere. En effet, il représente l'interface de communication entre les deux applications.

— ***Description des principaux cas d'utilisation***

❖ *Cas d'utilisation << Préciser l'url du service web >>*

Use case ***« Préciser l'url du service web »***
Pré-conditions : ▪ Un service web « PersonnalizeSystem » est déjà déployé.
▪ L'utilisateur consulte le système ESPERE. ▪ Il demande le changement de l'url actuel. ▪ Il donne la nouvelle adresse du service web. ▪ ESPERE vérifie la validité de l'url. ▪ Le système enregistre la nouvelle adresse dans le fichier de configuration. ▪ ESPERE recharge le fichier pour réinitialiser la connexion avec le service web disponible.
Exception *:* ▪ L'url donnée est non disponible ou elle sémantiquement incorrecte.
Extension *:* ▪ L'utilisateur garde l'ancienne url et quitte le système.
Post-conditions :

▪ Fichier url mit à jour avec la nouvelle adresse du service web disponible à l'application.

II.4. Package « Administration et configuration d'ESPERE»

– ***Cas d'utilisations du package « Administration et configuration du système ESPERE»***

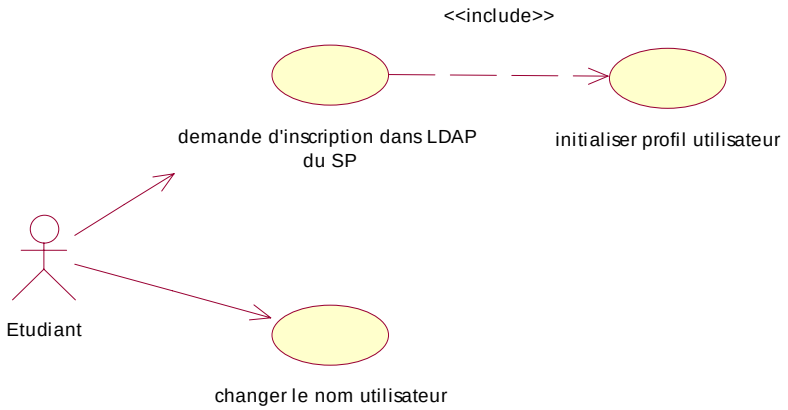


Figure 5. Diagramme des cas d'utilisation du package Configuration du système ESPERE.

Le package « Configuration du système ESPERE » récapitule les fonctionnalités d'authentification des utilisateurs et de la gestion de la correspondance entre profils utilisateurs et l'annuaire LDAP.

– ***Description des principaux cas d'utilisation***

❖ *Cas d'utilisation << demande d'inscription dans l'annuaire LDAP de ESPERE>>*

Use case *« demande d'inscription dans l'annuaire LDAP de ESPERE»*
Pré-conditions : Fichier de configuration contenant l'URL du service web disponible.
L'utilisateur consulte le système ESPERE. Il demande de s'inscrire dans l'annuaire LDAP. ESPERE vérifie que le nom utilisateur est enregistré. ESPERE envoie une requête de demande d'inscription au service web. Le service web transmet la requête vers ESPERE. ESPERE envoie une requête vers l'annuaire LDAP. Un accusé d'enregistrement est envoyé vers ESPERE.

Exception : Le service web est introuvable à l'adresse précisée. L'annuaire LDAP n'est pas disponible.
Extension : Le nom de l'utilisateur est un nom double.
Post-conditions :

III. Analyse et conception

III.1. Package « Recherche bibliothécaire»

I.1.1 Diagramme des classes

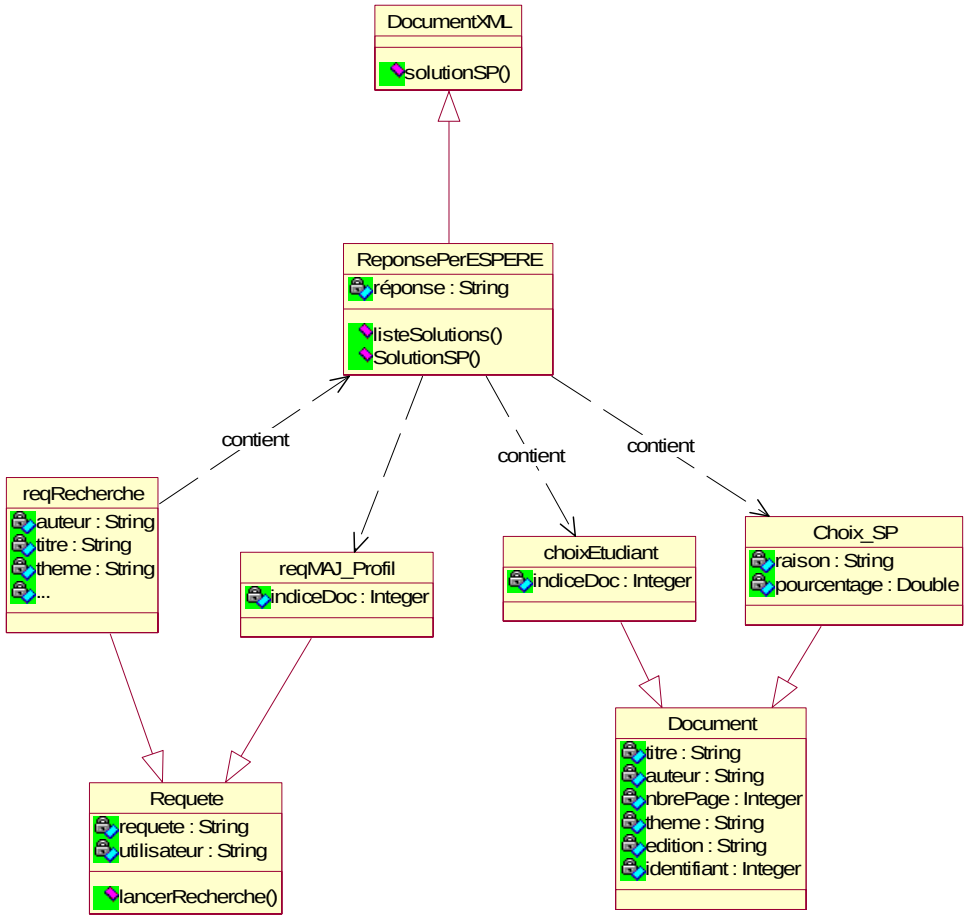


Figure 6. Diagramme de classe du package Recherche bibliothécaire.

I.1.2 Description des méthodes principales

- **lancerRecherche()** : méthode invoquée par l'étudiant qui lance une recherche de documents.
- **solutionSP()** : méthode invoquée lors de la réception d'une réponse à une requête de recherche pour extraire le stage préféré proposée par SP.
- **listeChoix()** : méthode invoquée pour extraire à partir de la réponse du SP la liste de tous les documents trouvés après l'exécution de la requête.

I.1.3 Diagrammes de séquences

❖ *Cas d'utilisation << Recherche personnalisée de documents >>*

Ce diagramme de séquence permet la recherche de documents qui répond aux préférences de l'étudiant en se basant sur les historiques, en effet cette recherche se base essentiellement sur les sujets par lesquels l'étudiant concerné est motivé.

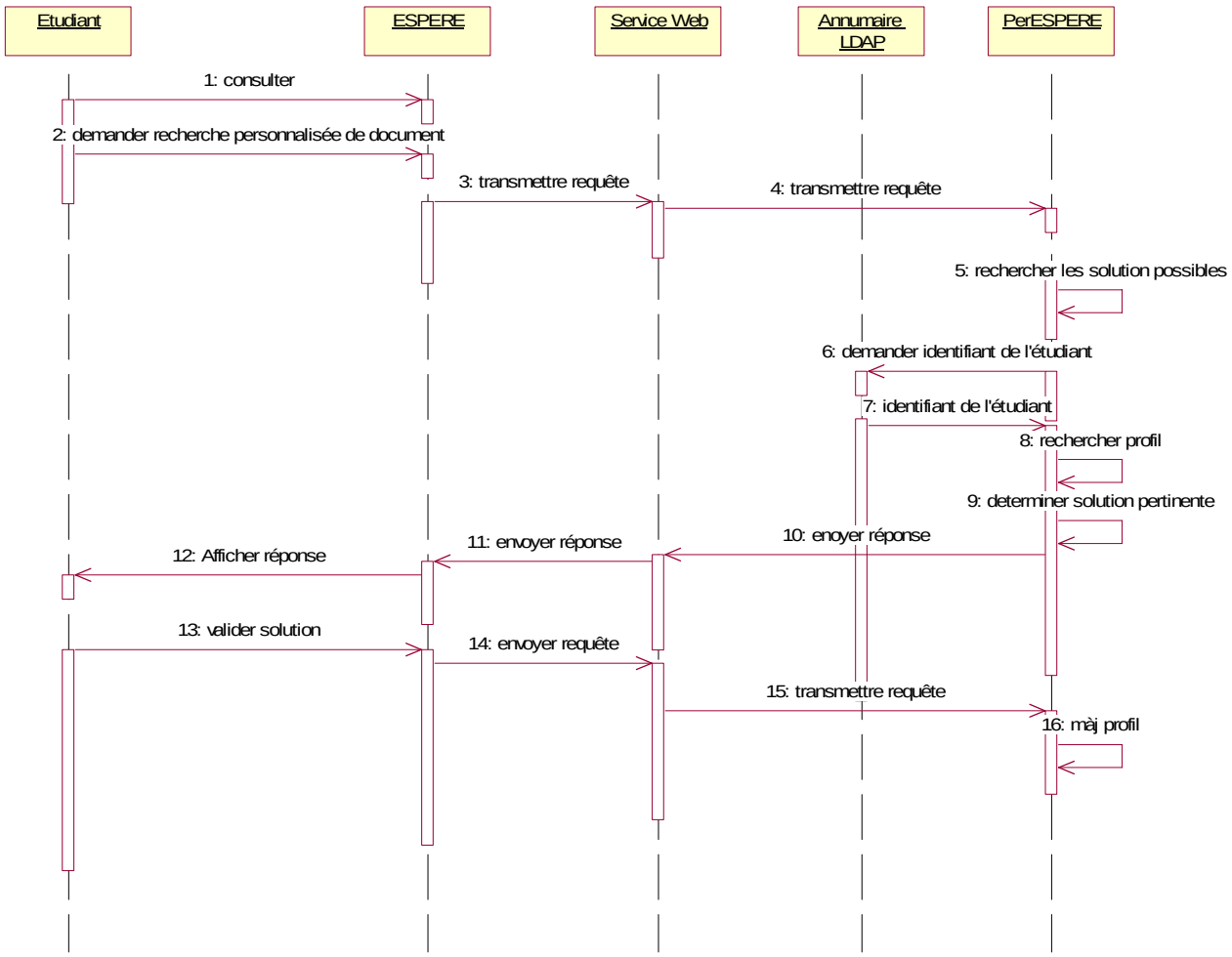


Figure 7. Diagramme de séquence « recherche personnalisée de documents ».

III.2. Package « Modules »

I.1.1 Diagramme des classes

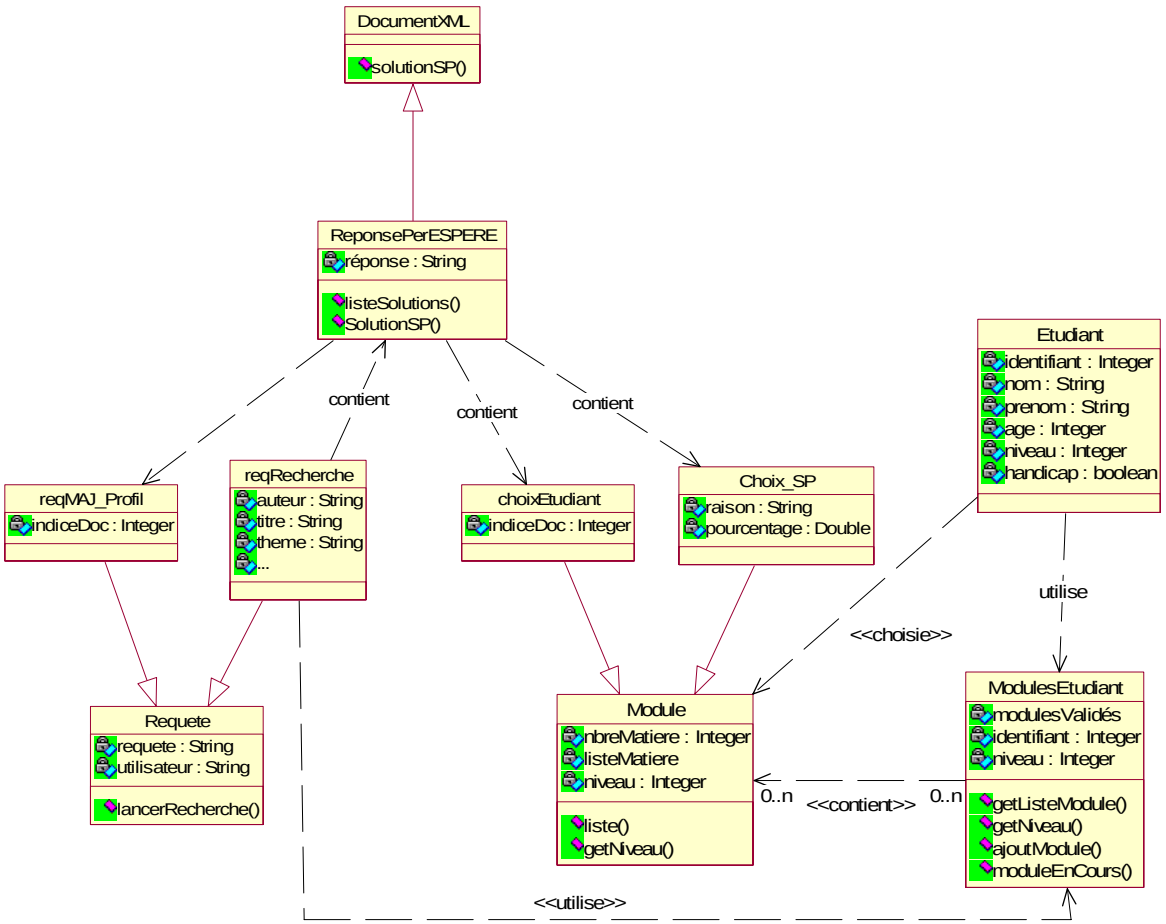


Figure 8. Diagramme de classe du package Modules.

I.1.2 Description des méthodes principales

- **lancerRecherche()** : méthode invoquée par l'étudiant pour lancer une requête de recherche de module personnalisée.
- **getListeModule()** : méthode invoquée lors de la recherche de module pour s'informer sur la liste des modules validés par l'étudiant,. En plus cette méthode peut être invoquée directement par l'étudiant pour connaître les modules qui à déjà validés.
- **ajoutModule()** : permet d'ajouter un module à la liste des modules choisis par l'étudiant, cette méthode invoque automatiquement la mise à jour du profil de l'étudiant.

5.3.7. Package « Service web "PersonnalizeSystem"»

5.3.7.1. Diagramme des classes

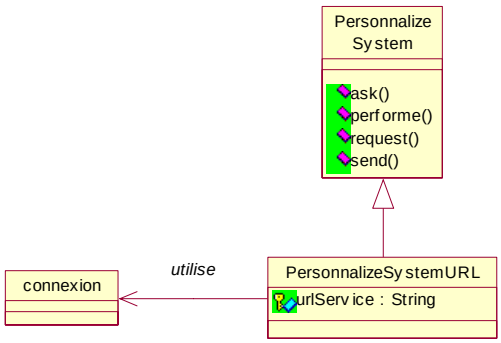


Figure 9. Diagramme de classe du package Service web « PersonnalizeSystem ».

5.3.7.2. Description des méthodes principales

- **ask()** : Pose une question au system de personnalisation
- **performe()** : Déclenche un événement au system de personnalisation
- **request()** : Lance une requête au system de personnalisation
- **send()** : Envoie un message au system de personnalisation

Diagrammes de séquences

❖ *Cas d'utilisation << Préciser l'url du service web >>*

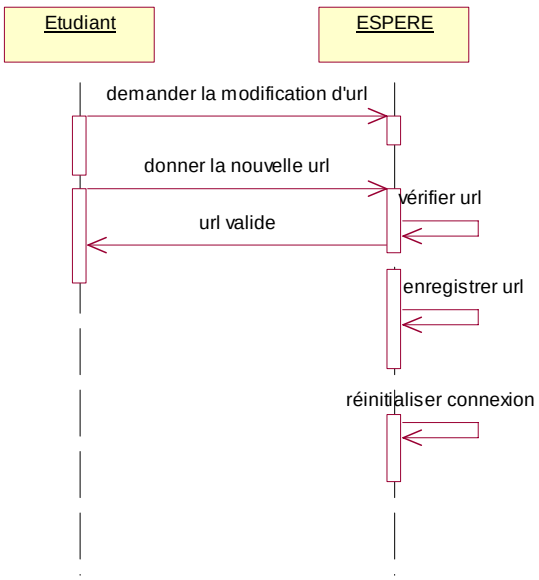


Figure 10. Diagramme de séquence « Préciser l'url du service web».

5.3.8. Package « Administration et configuration de ESPERE»

5.3.8.1. Diagramme des classes

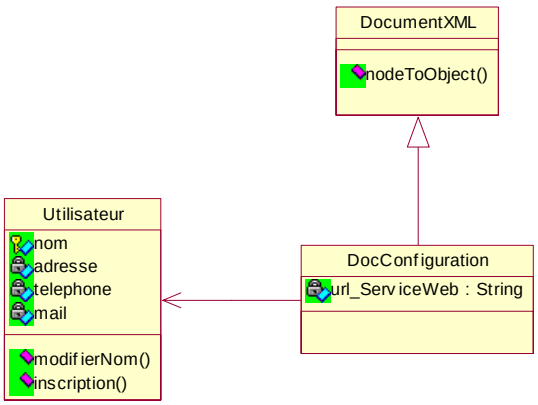


Figure 11. Diagramme de classe du package Administration et configuration de ESPERE.

5.3.8.2. Description des méthodes principales

- **modifierNom()** : cette méthode est invoquée lorsque l'étudiant décide de changer son idantifiant. Elle appel l'annuaire LDAP pour mettre à jour le nom d'inscription de l'utilisateur et appel EPSERE pour prendre en considération le nouveau nom pour la gestion du profil de l'utilisateur.
- **inscription()** : lors de la première connexion à PerEspere l'étudiant doit s'inscrire à l'annuaire LDAP. C'est l'invocation de cette méthode qui permet l'extraction du nom de l'étudiant à partir des informations personnelles enregistrées dans la plateforme d'accueil avec laquelle l'étudiant se connecte au système.

5.3.8.3. Diagrammes de séquences

❖ *Cas d'utilisation << demande d'inscription dans l'annuaire LDAP de PerEspere >>*

Ce diagramme de séquence permet à l'utilisateur de s'inscrire dans l'annuaire LDAP de PerEspere en passant par le service web pour l'envoie de la requête et en utilisant le nom utilisateur préalablement définie dans les informations personnelles comme identifiant unique de la plateforme d'accueil.

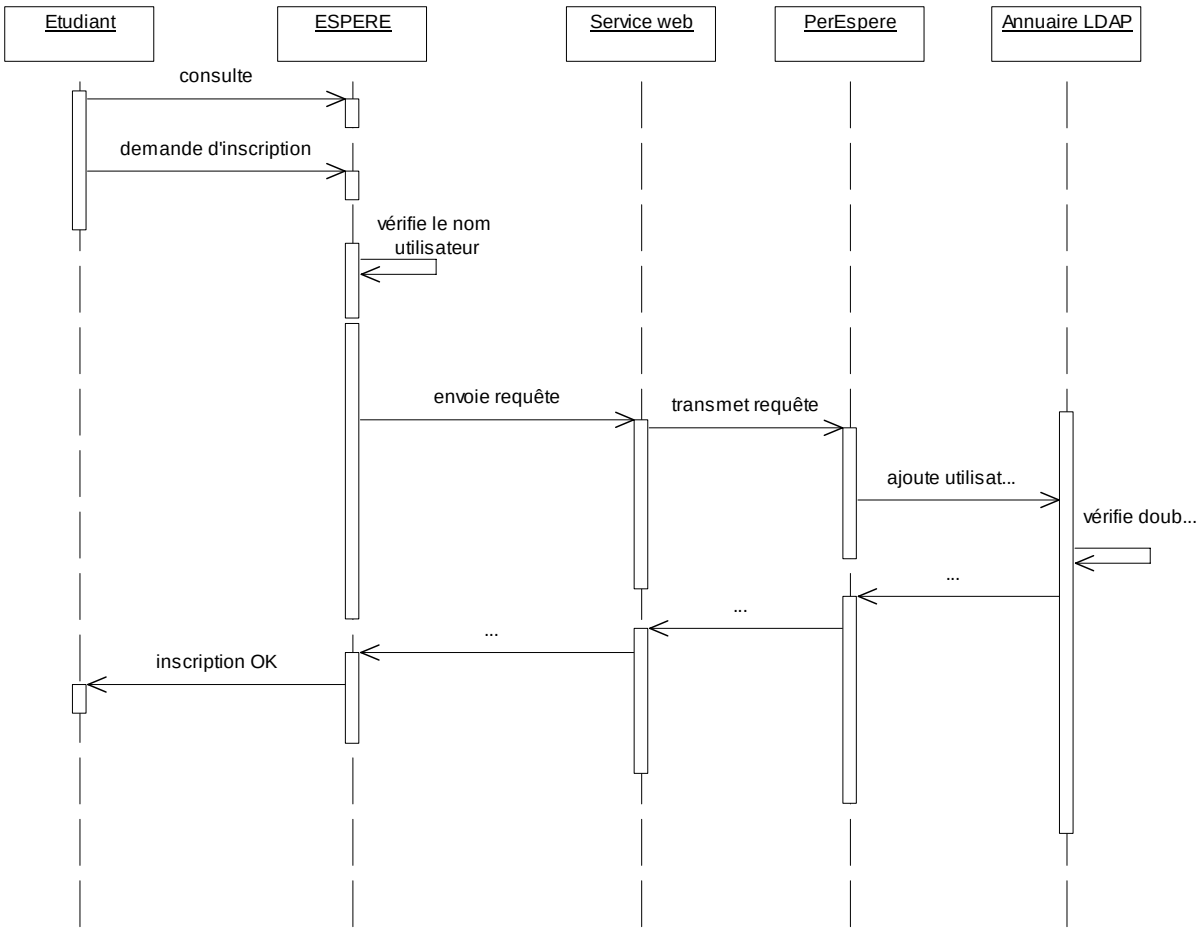


Figure 12. Diagramme de séquence « demande d'inscription LDAP ».

Lors de l'étape d'analyse et de conception, nous avons présenté les diagrammes de classes et leurs diagrammes de séquences. Par la suite, nous passons à la réalisation du système par priorité fonctionnelle.

IV. Environnement de développement

IV.1. La plate-forme « .NET »

« .NET » est la plate-forme Microsoft pour la nouvelle génération de logiciels distribués et coopérants, les services Web XML. Elle vise à simplifier la vie de l'utilisateur en lui fournissant des services intégrés, centrés sur lui, accessibles depuis tous ses périphériques, à tout moment et en tout lieu. Fondée sur des standards de l'industrie (http, XML, SOAP, WSDL), la plate-forme « .NET » est un moyen simple pour normaliser la coopération des services logiciels entre eux (voir l'architecture de .NET sur la figure 5.1).

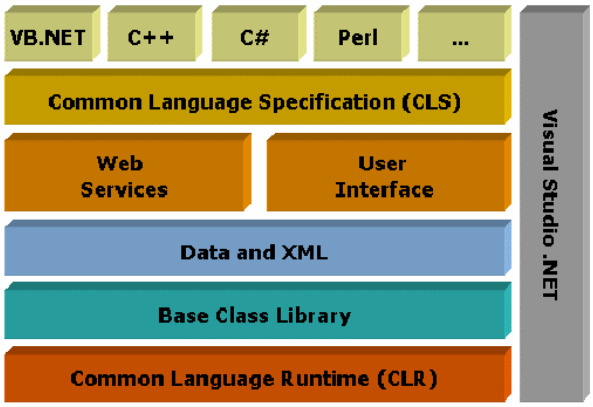


Figure 13. L'architecture fonctionnelle de .NET.

IV.2. Rational Rose

Rational rose, l'outil visuel principal de modélisation, permet de définir et de communiquer une architecture de logiciel aboutissant à un développement accéléré, à une qualité améliorée et à une visibilité augmentée.

Cet outil est utilisé pour éditer les différents diagrammes du modèle UML d'un logiciel, il permet de sauvegarder et d'imprimer ces diagrammes.

IV.3. Tomcat, Apache AXIS et OpenLDAP

- OpenLDAP11 est un projet libre de serveur d'annuaire conforme à la norme LDAP. Ce serveur entièrement gratuit et ces sources sont disponibles.
- Axis est un package qui fournit :
  - un environnement pouvant soit fonctionner comme un serveur SOAP indépendant soit comme un plug-in de moteurs de servlet (en particulier Tomcat),
  - Une API pour développer des services web SOAP RPC ou à base de messages SOAP.
  - le support de différentes couches de transport : HTTP, FTP...
  - la sérialisation/désérialisation automatique d'objets Java dans des messages SOAP.
  - des outils pour créer automatiquement les WSDL correspondant à des classes Java ou inversement pour créer les classes Java sur la base d'un WSDL.
  - des outils pour déployer et tester des web-services.
- Le serveur Tomcat est un serveur libre qui agit comme un conteneur de servlet J2EE. Tomcat implémente les spécifications des servlets et des JSP. Comme Tomcat inclut un serveur HTTP interne, il est aussi considéré comme un serveur HTTP.

[11] http://www.openldap.org/

Bibliographie générale

Bibliographie générale

[Aas97] K. Aas. *A Survey on Personalised Information Filtering Systems for the World Wide Web*. Research Report n 922, Oslo, Norway, Norwegian Computing Center, December 1997.

[Allen97] R. B. Allen. Mental Models and User Models. In M. Helander, T. Landauer, and P. Prabhu (Eds.), *Handbook of Human-Computer Interaction*, pp. 49–63. Elsevier Science B.V., 1997.

[Barron et al.99] T. M. Barron, R. H. Chiang, and V. C. Storey. A semiotics framework for information systems classification and development. *Decision Support Systems and Electronic Commerce*, 25(1) : 1–17, February 1999.

[Basu et al.98] C. Basu, H. Hirsh, and W. Cohen. Recommendation as Classification : Using Social and Content-Based Information in Recommendation. In C. Rich and J. Mostow (Eds.), *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98)*, pp. 714–720. Madison, USA, AAAI Press/MIT Press, July 26-30 1998.

[Beguin P. and A.Weill-Fassin97] *La simulation en ergonomie: Connaitre, Agir et interagir, Octares*.

[Billsus et al.99] D. Billsus and M. Pazzani. A Hybrid User Model for News Story Classification. In J. Kay (Ed.), *User Modeling : Proceedings of the 7th International Conference (UM'99 – Banff, Canada, June 20-24)*, pp. 99–108. Wien New York, Springer-Verlag, 1999.

[Boughzala01] I. Boughzala. *Démarche méthodologique de conception de systèmes d'information coopératifs interagent pour la gestion des connaissances*. Paris, France, Thèse de doctorat, Université de Paris VI, décembre 2001.

[Calvary et al.02] G. Calvary and J. Coutaz. Plasticité des interfaces : une nécessité ! In J. Le Maître (Ed.), *Actes des deuxièmes assises nationales du GDR I3*, pp. 247–261. Toulouse, France, Cépaduès, décembre 2002.

[Chau et al.03] M. Chau, D. Zeng, H. Chen, M. Huang, and D. Hendriawan. Design and evaluation of a multi-agent collaborative web mining system. *Decision Support Systems*, 35(1) : 167–183, April 2003.

[Chen et al.98] L. Chen and K. Sycara. WebMate : A Personal Agent for Browsing and Searching. In *Proceedings of the 2nd International Conference on Auto-nomous Agents and Multi Agent Systems*, pp. 132–139. Minneapolis, MN USA, ACM Press, May 10-13 1998.

[Cardon 00] Cardon A. *Conscience artificielle et systèmes adaptatifs*. Paris : Eyrolles, 2000.

[Goldberg 94] Goldberg D. Algorithmes génétiques : exploration, optimisation et apprentissage automatique. Paris : Addison Wesley, 1994.

[D'Aloisi et al.95] D. D'Aloisi and V. Giannini. The Info Agent : an Interface for Supporting Users in Intelligent Retrieval. In C. Stephanidis (Ed.), *Proceedings of th ERCIMWorkshop "Towards Interfaces for all : Current Trend and Future Efforts" (UI4ALL – Crete, October 30-31)*, pp. 143–155. 1995.

[David et al.01] A. David and F. Pallez. Les systèmes d'information à l'épreuve de l'organisation. In C. Cauvet and C. Rosenthal-Sabroux (Eds.), *Ingénierie des systèmes d'information*, pp. 23–60. Paris, France, Hermès, 2001.

[Davis et al.86] G. Davis, M. Olson, J. Ajenstat, and J.-L. Peaucelle. *Systèmes d'information pour le management*. Paris, Economica, 1986.

[De Rosnay75] J. De Rosnay. *Le macroscope : vers une vision globale*. Paris, Seuil, 1975.

[Dejean P.H.92] "*L'ergonomie du produit."* Performances n°57: pp 9-17.

[DeMontmollin M.95] Vocabulaire de l'ergonomie.

[Garrigou A., J. F. T., M.Jackson, F.Mascia 01] "*Contribution et démarche de l'ergonomie dans les processus de conception.*" PISTES vol 3(n°2).

[Goldberg et al.92] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35 : 61–70, 1992.

[Goecks et al.00] J. Goecks and J. Shavlik. Learning Users' Interests by Unobtrusively Observing Their Normal Behavior. In *2000 International Conference on Intelligent User Interfaces (IUI'00 – New Orleans, USA, January 9-12)*, pp.129–132. ACM, 2000.

[Gras A. , Joerges B., et al.92] *Sociologie des techniques de la vie quotidienne*. Paris, Edition L'harmattan.

[Grislin-Le Strugeon et al.01] E. Grislin-Le Strugeon, E. Adam, and C. Kolski. Agents intelligents en interaction Homme-Machine dans les Systèmes d'information. In C. Kolski (Ed.), *Interaction Homme-Machine pour les SI – Volume 2*, pp. 209–248. Paris, Hermès, 2001.

[Howden et al., 01] Howden N., Rönnquist R., Hodgson A. and Lucas A. JACK Intelligent Agents – Summary of an Agent Infrastructure. *5th International Conference on Autonomous Agents, Montreal*, Canada, May 2001.

[ITS, C.02] "*Pour un développement de l'information multimodale en agglomération : freins et perspectives*."

[Kadima et Monfort 03] Kadima H. et Monfort V. *Les web services : Techniques et outils XML, WSDL, SOAP, UDDI, Rosetta, UML*. Paris : Dunod, 2003.

[Kay95] J. Kay. The um toolkit for cooperating user modelling. *User Modeling and User-Adapted Interaction*, 4(3) : 149–196, 1995.

[Keeble et al.00] R. Keeble and R. Macredie. Assistant agents for the world wide web intelligent interface design challenges. *Interacting with Computers*, 12(4) : 357–381, February 2000.

[Kobsa et al.95] A. Kobsa and W. Pohl. The User Modeling Shell System BGP-MS. *User Modeling and User-Adapted Interaction*, 4(2) : 59–106, 1995.

[Kobsa01] A. Kobsa. Generic User Modeling Systems. *User Modeling and User-Adapted Interaction*, 11 : 49–63, 2001.

[Krulwich et al.97] B. Krulwich and C. Burkey. The InfoFinder agent : Learning User Interests through Heuristic Phrase Extraction. *IEEE Expert : Intelligent Systems and Their Applications*, 12(5) : 22–25, September/October 1997.

[Kuflik et al.00] T. Kuflik and P. Shoval. User Profile Generation for Intelligent Information Agents – Research in Progress. In G. Wagner, Y. Lespérance, and E. Yu (Eds.), *Agent-Oriented Information Systems (AOIS'00 – Stockholm,June 5-6)*. Berlin, Germany, iCue Publishing, 2000.

[Lapointe93] J. Lapointe. L'approche systémique et la technologie de l'éducation. EducaTechnologiques, 1(1), février 1993.

[Lashkari et al.97] Y. Lashkari, M. Metral, and P. Maes. Collaborative interface agents. In M. Huhns and M. Singh (Eds.), *Readings in Agents*, pp. 111–116. Morgan Kaufmann, 1997.

[Lieberman95] H. Lieberman. Letizia : An Agent That Assists Web Browsing. In *International Joint Conference on Artificial Intelligence (IJCAI'95 – Montreal, Canada, August 20-25)*. San Francisco, USA, Morgan Kaufmann Publishers, 1995.

[Lieberman97] H. Lieberman. Autonomous interface agents. In *Proceedings of CHI'97 (Human factors in computing systems)*, pp. 67–74. Atlanta, GA USA, ACM Press, March 22-27 1997.

[Manber et al.00] U. Manber, A. Patel, and J. Robison. Experience with personalization on Yahoo ! *Communications of the ACM*, 43(8) : 35–39, August 2000.

[Mandiau et al., 02] Mandiau R., Grislin-Le Strugeon E. et Péninou A. (Ed.). Organisation et applications des SMA. Paris : Hermès, 2002.

[Michel et al.99] C. Michel and S. Lainé-Cruzel. Profil-Doc : Un prototype de système de recherche d'information personnalisé selon le profil des utilisateurs. In *Workshop "Documents virtuels personnalisables" (IHM'99 – Montpellier, France, 22-26 novembre)*. 1999.

[Minati] G. Minati. *Introduction à la systémique*. http ://www.afscet.asso.fr/resSystemica/.

[Rolland99] C. Rolland. État de l'art et perspectives. Cours de DEA, Module "Conception des systèmes d'information", 1998/1999.

[Nwana96] H. S. Nwana. Software Agents : An Overview. *Knowledge University Review*, 11(3) : 1–40, September 1996.

[Orwant93] J. Orwant. *Doppelgänger Goes To School : Machine Learning for User Modeling*. PhD. Thesis, Massachussetts Institute of Technology, September 1993.

[Pache, Y. 02] L'information sur les perturbations dans les transports publics urbains de surface : analyse et perspectives de développement. PARIS, ENTPE.

[Pavard B.,02] "*RTP Acceptabilité, ergonomie et usages*." CNRS STIC textes de cadrage.

[Peretz H.98] *Les méthodes en sociologie: L'observation*. Paris.

[Pohl et al.97] W. Pohl and J. Höle. Mechanisms for flexible representation and use of knowledge in user modeling shell systems. In A. Jameson, C. Paris, and C. Tasso (Eds.), *User Modeling : Proceedings of the Sixth International*

Conference (UM'97 – Sardinia, Italy, June 2-5), pp. 403–414. Wien New York, Springer-Verlag, 1997.

[Rey00] A. Rey (Ed.). *Dictionnaire Historique de la langue Française*. Paris, Le Robert, 2000, 3ème édition.

[Reix00] R. Reix. *Systèmes d'information et management des organisations*. Paris, France, Vuibert, août 2000, 3ème édition.

[Reix99] R. Reix. *Dictionnaire des systèmes d'information*. Paris, France, Vuibert, novembre 1999.

[Rizcallah 00] Rizcallah M. *Construire un annuaire d'entreprise avec LDAP*. Paris : Eyrolle, 2000.

[Routier et al., 01] Routier J.C., Mathieu P. and Secq Y. Dynamic skill learning : A support to agent evolution. In *Proceedings of the AISB'01 Symposium on Adaptive Agents and Multi-Agent Systems*, pp. 25-32, University of York, United Kingdom, March 2001.

[Rozé et al.00b] C. Rozé, E. Grislin-Le Strugeon, M. Abed, G. Uster, and C. Kolski. Recherche d'informations personnalisées. In *Conférence Internationale NîmesTIC 2000 Ingénierie des Systèmes et NTIC (NTIC'00 – Nîmes,France, 9-11 septembre)*, pp. 401–407. 2000.

[Salton et al.83] G. Salton and M. McGill. *Introduction to modern information retrieval*. New York, McGraw-Hill, 1983.

[Shardanand et al.95] U. Shardanand and P. Maes. Social Information Filtering : Algorithms for Automating 'World of Mouth'. In *Proceedings of the CHI-95 Conference*. Denver, CO USA, ACM Press, May 1995.

[Silvestre et al.94] P. Silvestre and D. Verlhac. Stratégie de conception des systèmes d'information. *Techniques de l'ingénieur*, juin 1994.

[Thevenin01] D. Thévenin. *Adaptation en Interaction Homme-Machine : le cas de la Plasticité*. Grenoble, Thèse de doctorat, Université Joseph Fourier, décembre 2001.

[Urso97] P. Urso. Identifiez les moteurs adaptés à vos besoins, et travaillez off-line. *Technologies internationales*, 38 : 42–45, octobre 1997.

[Virvou et al.99] M. Virvou and B. Du Boulay. Human plausible reasoning for intelligent help. *User Modeling and User-Adapted Interaction*, 9(4) : 321–375, 1999.

[Waern et al.98] A. Waern, M. Tierney, Å. Rudström, and J. Laaksolahti. *ConCall : An Information service for researchers based on EdInfo*. Research Report n ° T98:04, Swedish Institute of Computer Science, October 1998.

[Waern et al.99] A.Waern, M. Tierney, Å. Rudström, and J. Laaksolahti. ConCall : Edited and Adaptive Information Filtering. In *1999 International Conference on Intelligent User Interfaces (IUI'99 – Los Angeles, USA, January 5-8)*, p.185. ACM, 1999.

[Yen et al.02] B. Yen and R. Kong. Personalization of information access for electronic catalogs on the web. *Electronic Commerce Research and Applications*, 1 : 20–40, 2002.

[Zreik K .94] "*Organisation de la conception*." Editions EUROPIA.