Université de Montréal

Faculté des Études Supérieures

Ce mémoire intitulé:

Architecture de Bibliothèque Numérique Multi-agents À Services Extensibles

Présenté par:

Kamel Hamard

a été évalué par un jury composé des personnes suivantes:

Président-rapporteur : **Jean Vaucher**

Directeur de recherche : **Jian-Yun Nie**

Codirectrice de recherche : **Brigitte Kerhervé**

Membre du jury : **Esma Aimeur**

Mémoire accepté en Mars 2000

Université de Montréal

Architecture de Bibliothèque Numérique Multi-agents À Services Extensibles

Présenté par

Kamel Hamard

Département d'Informatique et Recherche Opérationnelle

Faculté des Arts et des Sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique

Décembre, 1999

SOMMAIRE

L'information sous toutes ses formes (image, son, texte) a pris une place prépondérante dans la société d'aujourd'hui. Elle est devenue indispensable et chacun exige de pouvoir y accéder de manière simple, efficace et rapide. L'évolution rapide du monde de l'*Internet* a permis essentiellement de rendre disponible toute cette masse d'informations. Cependant le partage et l'accès à cette information devient difficile au fur et à mesure qu'elle augmente. Les recherches entreprises ont permis de faire face à ce défi en mettant au point différents systèmes qui intègrent des bases d'informations de natures variées et distribuées. Par cette intégration, un usager peut interroger toutes les bases d'informations de façon uniforme et transparente. Une bibliothèque numérique distribuée est un système qui réalise cette fonction.

Ce mémoire décrit les résultats des travaux de recherche qui nous ont permis de proposer une architecture présentant des solutions à certains des problèmes les plus cruciaux à savoir: l'extensibilité, l'expansion et l'organisation du stockage des données. En effet, l'architecture proposée est une architecture qui organise le système de stockage en deux niveaux d'indexation. Ces niveaux ont pour principal objectif de permettre la prise en compte d'une masse importante de données disséminées à travers le réseau Internet et de rendre son accès plus rapide et par la même occasion éviter l'encombrement du réseau. Quant au problème de l'extensibilité et de l'expansion, il a été résolu grâce à l'utilisation d'une architecture multi-agents orienté services.

REMERCIEMENTS

J'exprime ma profonde gratitude à mon directeur de recherche, le professeur Jian-Yun Nie, et à ma codirectrice, la professeure Brigitte Kerhervé du département d'informatique de l'université du Québec à Montréal, pour avoir dirigé avec patience, confiance et enthousiasme ce mémoire. Le support financier et la disponibilité qu'ils m'ont accordés ont été un grand stimulant pour mener à terme mes travaux de recherche.

De même, je remercie tous les membres de ma famille et en particulier mon épouse qui n'a cessé de me soutenir tout au long de mes études.

Table des Matières

LISTE DES FIGURES

Chapitre 1

Introduction

L'évolution rapide du monde de l'*Internet* a permis de mettre à la disposition d'une communauté d'usagers de plus en plus importante, de grandes masses d'informations numériques (textes, images et sons). Cependant, le problème de gestion de l'accès et du partage de l'information devient difficile au fur et à mesure que cette masse d'informations augmente. Les recherches entreprises ont permis de mettre au point certaines techniques permettant de rechercher différents types de documents, c'est-à-dire des documents textuels, des documents images ou sonores.

Un des thèmes de recherche vise à développer des systèmes permettant d'intégrer des bases d'informations de natures variées et distribuées. Par cette intégration, un usager peut interroger toutes les bases d'informations de façon uniforme et transparente. Une bibliothèque numérique distribuée (BND) est un système qui réalise cette fonction. Ce mémoire décrit les résultats de nos travaux de recherche qui ont permis la mise en œuvre d'une BND.

1.1 Définition du problème

Une bibliothèque numérique est un système qui fournit, à une communauté d'usagers, un accès à une large collection d'objets numériques associés à différents artefacts (livres, images, vidéo, etc.). Par bibliothèque numérique distribuée (BND) nous désignons un système qui intègre plusieurs collections d'objets localisées à différents endroits. Cette

intégration permet aux usagers d'accéder, de manière transparente, aux collections sans se préoccuper de leurs particularités (langue, localisation, etc.), voir Figure 1.1.

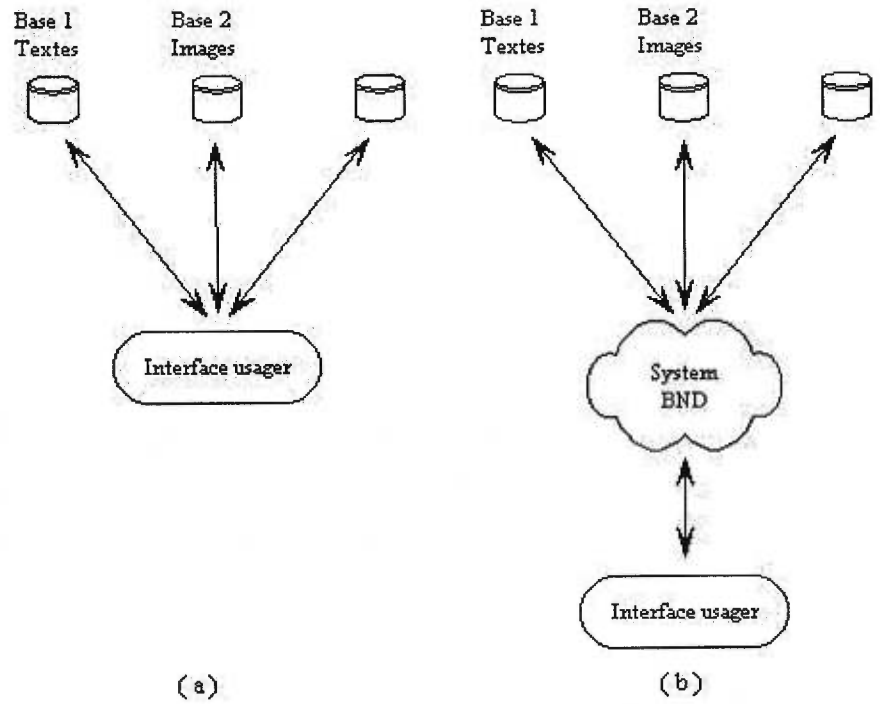


Figure 1.1 : (a) accès directe aux collections par l'usager. (b) accès via une BND.

Plusieurs projets de réalisation de BND ont été menés ces dernières années et les aspects abordés par ces recherches concernent :

- La description des objets numériques et des banques de données,
- La gestion et l'organisation des collections,
- L'extensibilité des systèmes de bibliothèques numériques distribuées (BND),
- L'interopérabilité dans un environnement de BND,
- L'interface usager et l'interaction homme-machine.

Cependant, plusieurs problèmes n'ont pas encore été traités de façon satisfaisante, comme par exemple l'architecture des systèmes, l'accès aux données multimédia et

l'interrogation des données dans différentes langues. Cette étude tente d'apporter certains éléments de réponse à ces problèmes, notamment :

- La définition de l'architecture globale d'un système de bibliothèque numérique,
- L'extensibilité de l'architecture d'une bibliothèque numérique,
- L'organisation des services offerts aux usagers,
- L'organisation et gestion de l'accès aux collections multimédia.

Nous proposons une solution globale qui est implantée sous la forme d'un prototype afin de mettre en œuvre les différents concepts introduits tout au long de l'étude.

1.2 Cadre de réalisation

Notre projet de maîtrise s'insère dans le cadre du projet de recherche « Distributed Multimedia Digital Librairies » financé par le Fond de Développement de l'Autoroute de l'Information (FADI) du gouvernement du Québec et Newbridge. Le principal objectif du projet est d'étudier les problèmes liés à la représentation et à l'accès aux informations multimédia dans un environnement distribué.

Quatre universités ont pris part à la réalisation du projet. L'université *Concordia*, l'université *McGill*, l'université de Montréal (*UdM*) et l'université du Québec à Montréal (*UQAM*). Ainsi, quatre sous-projets ont été définis :

1. Indexation et classification des documents visuels dans les bibliothèques numériques (UQAM),
2. Structuration et classification des objets multimédia tridimensionnels dans les bibliothèques numériques spécifiques aux environnements virtuels (McGill),
3. Utilisation d'agents intelligents, au niveau de l'interface usager, pour supporter l'accès aux informations contenues dans les bibliothèques numériques (Concordia),

4. Traitement de requêtes multilingues dans les bibliothèques numériques ainsi que la conception d'un prototype intégrant les composants de tous les sous-projets (UdM).

Cette étude s'inscrit dans le dernier volet de ce projet.

1.3 Objectifs de la Recherche

On distingue deux parties dans le projet de recherche réalisé. La première partie concerne la conception et l'implantation d'un système distribué de bibliothèque numérique multi-agents. Dans ce cas, nous avons opté pour une nouvelle architecture dans laquelle les agents sont organisés hiérarchiquement afin d'optimiser les opérations de recherche. La seconde est en rapport avec la représentation et l'accès aux documents multimédia. Pour ce faire, nous avons utilisé le standard Z39.50 qui permettra une plus grande accessibilité aux données.

La base d'images utilisée pour concevoir le prototype de démonstration du projet de bibliothèque numérique dans lequel s'insère ce projet de maîtrise est la collection "Ramsay Traquair : The Architectural Heritage of Québec", communément appelée base Traquair. La base Traquair fait partie de la collection canadienne d'architecture. Elle est constituée de 7.922 photographies numérisées d'édifices architecturaux du Québec et d'objets en argent ayant servi à des fins domestiques ou religieuses. Ces images, qui représentent une grande partie de l'histoire de l'architecture au Québec, proviennent principalement de l'œuvre de Ramsay Traquair: architecte, enseignant et directeur de l'école d'architecture de l'université McGill.

Nous nous sommes posés plusieurs questions durant les travaux de recherche effectués. Nous en résumons l'essentiel :

- Comment organiser les données des artefacts multimédia ?
- Comment organiser les informations utiles pour une intégration rapide et efficace de la recherche d'image selon des caractéristiques visuelles dans le standard Z39.50 ?

- Comment combiner les différents types de recherche (Image et texte) pour effectuer une recherche efficace ?
- Comment prendre en compte la multitude de bases de données différentes ?
- Comment gérer efficacement un flot continu de requêtes ?
- Comment assurer l'extensibilité du système ?

1.4 Apport de la Recherche

L'étude des problèmes abordés dans ce projet de recherche nous a permis de proposer des solutions aux problèmes posés en nous basant sur différentes technologies. Cette étude a permis d'atteindre les objectifs suivants:

- Montrer la pertinence et l'importance de l'extension du standard Z39.50 à la recherche d'images,
- Mettre en évidence les caractéristiques les plus importantes à prendre en compte lors de la conception d'une BND,
- Proposer une architecture basée sur différents agents de service et par la même occasion montrer la flexibilité offerte par ce type de solution,
- Proposer un système d'indexation à plusieurs niveaux et montrer son intérêt pour la gestion de grandes quantités d'informations.

1.5 Organisation du Rapport

Le présent rapport est organisé en quatre chapitres. Le premier chapitre fait la présentation de la problématique de recherche, le second fait la description de l'état de l'art et introduit toutes les notions théoriques en rapport avec les aspects concernés par cette étude. Le troisième chapitre concerne la description de l'architecture multi-agents du système et la conception du prototype. Dans le quatrième chapitre nous détaillons toutes les décisions d'implantation du prototype en décrivant les systèmes et langages qui ont permis la mise en œuvre du système. Le cinquième et dernier chapitre permettra

de faire le point sur l'expérience acquise dans l'étude ainsi que les questions qui restent toujours ouvertes.

Chapitre 2

2 État de l'art

L'objectif d'une bibliothèque numérique distribuée est de permettre à différents usagers de partager des documents. Elle joue un rôle similaire à celui d'une bibliothèque conventionnelle, mais dans un environnement moderne, numérique et réseauté. Une bibliothèque conventionnelle contient généralement un grand nombre de documents de différentes natures (livres, revues, œuvres d'art, etc.). Dans une bibliothèque numérique distribuée il y a aussi une large collection de données multimédia. Ces données numériques peuvent être des documents traditionnels numérisés ou créés numériquement. En plus de ces données, un système de bibliothèque numérique distribuée fournit aussi les moyens d'organiser, gérer et accéder aux informations numériques. Ce n'est pas seulement un outil de recherche d'informations mais plutôt un ensemble de fonctionnalités et de services qui vont permettre de partager, gérer, interpréter, découvrir, chercher et résumer des informations.

Les recherches sur la BND visent à rendre un tel système possible, elles ont abordé les aspects suivants :

Description des objets numériques: tout artefact doit avoir une représentation numérique interne. Cette représentation doit permettre, selon le besoin, des recherches simples ou sophistiquées. Nous pouvons citer à titre d'exemple le cas des images pour lesquelles toutes les informations qui identifient les objets dans l'image doivent avoir été extraites afin de permettre des recherches sur le contenu.

Organisation et traitement de données multimédia: l'organisation et le traitement des données revêt une grande importance vu l'influence qu'ils exercent sur les performances du système. C'est le cas du stockage et l'accès aux données.

L'interface d'interrogation usager: l'interface d'interrogation est un aspect non négligeable dans la conception d'un système destiné au grand public. Celle-ci doit permettre l'interrogation d'une ou plusieurs collections le plus simplement possible et doit offrir le moyen d'exprimer des besoins complexes.

L'extensibilité des systèmes de BND: cette propriété doit permettre l'intégration de nouveaux services et fonctionnalités au système de manière simple, rapide et efficace et ceci pour faire face aux besoins des usagers. Cette propriété est particulièrement importante dans l'état actuel où les recherches sur les BND n'ont pas atteint une maturité suffisante.

L'expansion des systèmes de BND: il est très important que la masse de données, qui enrichissent continuellement le système, soit prise en compte sans affecter les performances.

L'interopérabilité: l'interopérabilité permet d'assurer les échanges entre les systèmes existants. Ainsi, les usagers ont accès, de manière transparente, à une multitude de bases de données qui sont desservies par différents systèmes.

Plusieurs systèmes de BND ont été décrits dans la littérature. Ils offrent la possibilité de rechercher des documents de type textuel ou image. Parmi les techniques utilisées pour la recherche d'images, il y a celles qui se basent sur les annotations textuelles et celles qui se basent sur les caractéristiques visuelles des images [Chang 97]. Les caractéristiques visuelles sont rajoutées pour compléter les techniques textuelles afin d'améliorer de manière significative les possibilités de recherche.

Un ensemble de services a été observé dans une large majorité des systèmes étudiés. Certains sont considérés comme étant des services de base alors que d'autres sont plus spécifiques aux applications.

2.1 Les services de base

Parmi les services de base identifiés par [Grossman 95] nous avons :

- Les services de stockage,
- Les services de recherche,
- Les services de navigation.

Le service de stockage fournit une interface permettant à des clients d'accéder à des documents. Il offre aussi une interface basée sur des transactions qui permettent aux archivistes de modifier ou d'ajouter des documents et aux clients de savoir à tout moment les changements qui se produisent dans la collection.

Le service de recherche permet à un usager de retrouver des documents en utilisant un système d'indexation. Des informations textuelles ou visuelles sont utilisées pour générer les index qui vont permettre d'effectuer des recherches dans la base d'images.

Le service de navigation est le lien entre le service de stockage et le service de recherche. Il permet à l'usager de naviguer dans la liste des documents retournés par la recherche.

2.2 Les Services d'Extension

En plus des trois services cités ci-dessus, il existe d'autres services tous aussi importants, ils se résument comme suit [Hylton 94]:

- Visualisation de documents,
- Indexation de documents,

- Conversion de formats d'un document,
- Utilisation de profil usager,
- Notification électronique des changements.

Dans le cadre de notre projet, seuls les trois premiers services ont implantés.

Un des objectifs principaux de ce mémoire est de proposer une architecture de système adaptée aux BNDs. Dans le reste de ce chapitre, nous allons d'abord analyser quelques systèmes existants afin de dégager des propriétés importantes dans l'architecture du système de BND désirée. Nous décrirons ensuite l'aspect de méta-information qui est primordial pour l'intégration des données multimédia. Nous présenterons finalement un standard de méta-information et d'échange d'informations Z39.50.

2.3 Systèmes de Bibliothèques Numériques Existants

Les systèmes d'informations multimédia doivent permettre le traitement, le stockage, la gestion, la recherche, le transfert et la présentation de documents d'une collection. L'efficacité et les performances de ces systèmes vont dépendre de la qualité des services offerts, du type de gestion de ces services et de l'organisation du stockage. Nous avons étudié les architectures les plus pertinentes, c'est-à-dire celles en rapport avec la problématique de la gestion de services et de l'organisation des données dans un environnement distribué.

2.3.1 Le projet ADL [Andressen 95]

Le projet Alexandria Digital Library (ADL) a permis la mise au point de l'une des premières bibliothèques numériques distribuées. ADL est constituée de plusieurs nœuds distribués à travers le réseau Internet. Chaque nœud supporte une composante de la librairie. Ces composantes étant la collection, le catalogue, l'interface et le peuplement de la base des images. La Figure 2.1 montre les composantes de l'architecture du prototype de ADL.

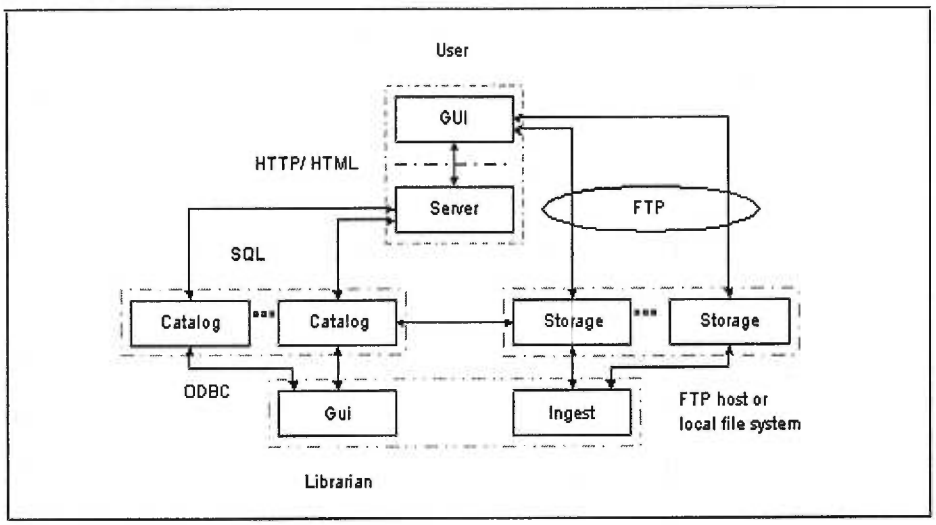


Figure 2.1 : Architecture de ADL

Les aspects les plus importants de cette architecture sont les suivants :

1) Le *stockage* des collections : les collections et les catalogues sont distribués à travers plusieurs nœuds.

2) Le *catalogue :* le catalogue stocke les données bibliographiques des documents. Il utilise un modèle de métadonnées extensible et il peut être consulté via un moteur de recherche dont les requêtes sont conformes au standard SQL/Z39.50[1]. L'utilisateur a l'accès au catalogue grâce aux deux protocoles HTTP et Z39.50. Deux types de connexions sont alors possibles: la connexion au serveur de ADL à l'aide d'un browser Web (protocole HTTP) et la connexion au service Z39.50 à travers un client Z39.50. Les requêtes, conformes au standard Z39.50, sont distribuées à l'ensemble des nœuds serveurs du système qui respectent ce protocole. Une liste de résultats unique est créée à partir des résultats renvoyés par les différents serveurs.

[1] Z39.50 est un standard du NISO (National Information Standard Organization).

3) L'*interface:* l'interface usager est réalisée en utilisant les standards HTTP/HTML, de fureteurs Web et d'afficheurs. Une fois que l'utilisateur soumet sa requête au moteur de requêtes, celui-ci la traduit au format des différents catalogues existant dans le système (SQL, ConQuest et Z39.50) et effectue la recherche dans les catalogues locaux et distants. Le moteur de recherche retourne ensuite les résultats à l'interface utilisateur. La liste des items retournés contient des métadonnées de type texte ou image. L'utilisateur peut sélectionner des informations complémentaires sur chacun des documents retournés.

Les bases contiennent essentiellement des documents images et les attributs de recherche dans le catalogue sont la texture, la couleur, l'histogramme et la forme des objets dans les images.

2.3.2 Le projet de Stanford University [Paepcke 96a]

L'objectif du projet de Stanford university était d'assurer une interopérabilité entre les *BNDs* permettant ainsi à un utilisateur d'accéder à n'importe quelle information, où qu'elle soit située [Paepcke 96a]. Les chercheurs ont proposé d'organiser tous les services possibles autour d'un bus commun conçu à l'aide de la technologie CORBA comme l'indique la Figure 2.2. Dans la nouvelle architecture, l'usager peut utiliser les protocoles de connexion Telnet, HTTP ou Z39.50 pour accéder à l'ensemble des services des librairies via Infobus. Cette nouvelle architecture se base sur l'utilisation de proxy pour accéder aux services en ligne. Un protocole expérimental a été développé pour permettre aux clients *CORBA* d'interagir avec les proxy de la bibliothèque numérique [Paepcke 96b].

Les principales caractéristiques de cette architecture sont la flexibilité et l'extensibilité qui vont permettre de gérer des modèles d'informations organisationnels complexes des services variés et de s'étendre facilement. Ces propriétés sont nécessaires pour les BND.

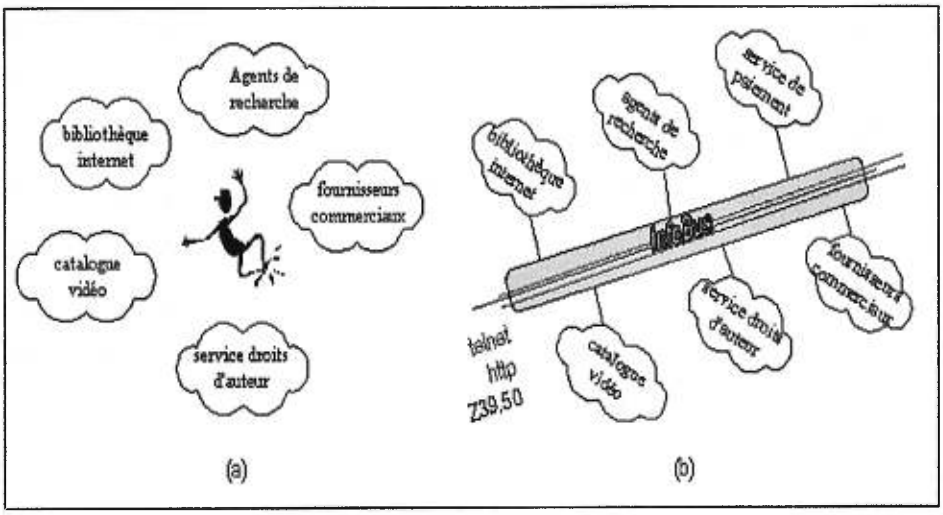


Figure 2.2 : (a) Services non organisés, (b) Services organisés autour de l'infobus.

Parmi les systèmes qui ont tenu compte des propriétés citées ci-dessus nous pouvons citer les systèmes Harvest et UMDL. Dans le premier cas nous avons un système basé sur plusieurs services. Dans le second cas, nous avons un système qui utilise une communauté d'agents pour assurer les fonctionnalités du système.

2.3.3 Le système Harvest

Harvest [Bowman 94] est un système qui dispose d'un mécanisme de collecte et de distribution d'information d'indexation sophistiqué. Ce mécanisme permet d'éviter la surcharge de serveurs et des connexions réseaux en utilisant différentes entités comme l'illustre la Figure 2.3. Ces entités sont:

1- Les courtiers (borker)

Ce sont les serveurs d'index de Harvest. Ils sont constitués de plusieurs éléments qui permettent de construire des index du système, grâce à la collecte auprès des différents collecteurs locaux ou d'autres courtiers. Ces éléments sont constitués, entre autres, d'un gestionnaire de requêtes et d'un gestionnaire de stockage et d'indexation.

2- *Le collecteur local*

C'est un élément que nous retrouvons au niveau d'un site de données. Sa fonction est de regrouper un résumé des données publiques du site afin de les exporter vers le courtier. Périodiquement, de nouvelles données sont constituées et stockées dans un cache local en attente d'être collectées par le courtier à des fins d'indexation.

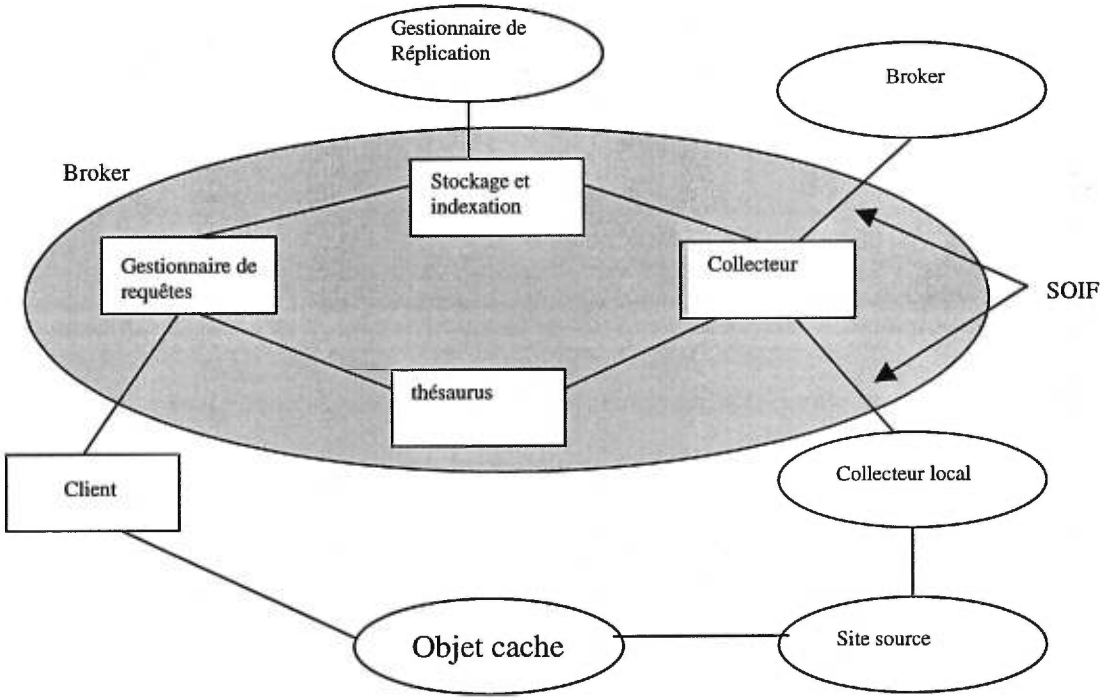


Figure 2.3 : Architecture de HARVEST.

3 - Le client

Il soumet une requête au courtier pour la recherche de documents. Ces derniers sont récupérés directement du site où ils résident. Un objet cache est utilisé pour mémoriser momentanément des documents retrouvés pour accélérer leur récupération.

4 - Le site source:

C'est le site où se trouvent les données à indexer. Plusieurs sites de ce type peuvent exister dans le système.

Nous observons dans ce système une certaine homogénéité:

- L'indexation des documents est faite par un mécanisme propre au système (dans le courtier). Ce n'est pas une opération effectuée par le fournisseur de l'information. Ainsi, nous pouvons nous attendre à une indexation plus uniforme.
- Des communications peuvent être initiées à tout moment entre les courtiers. Ce sont des composantes identiques qui se basent sur un protocole de communication interne SOIF (Summary Object Interchange Format).

Grâce à cette homogénéité, l'intégration du système s'en trouve simplifiée. Nous pouvons aussi remarquer l'efficacité du mécanisme d'indexation. En effet, celui-ci permet une meilleure utilisation du réseau en soumettant seulement le résumé des nouveaux documents à indexer. La surcharge des serveurs est évitée grâce à la participation intelligente des différents courtiers dans le processus d'indexation et de recherche.

2.3.4 Le projet UMDL [Birmingham 95]

UMDL marque une étape importante dans le développement des BND. Beaucoup de systèmes ultérieurs se sont inspirés de cette architecture. Dans le projet UMDL, développé par l'université du Michigan, les problèmes principalement traités sont l'extensibilité et l'expansion via l'utilisation de différents types d'agents [Birmingham 95]. Nous entendons par extensibilité la possibilité de créer facilement de nouveaux services et par expansion la capacité qu'a un système de se développer.

L'architecture du système UMDL est basée sur un ensemble d'agents dont les propriétés principales sont l'autonomie et la négociation. L'autonomie est la capacité qu'a un agent à raisonner et prendre des décisions quant à l'utilisation des ressources dont il dispose. L'autonomie implique une capacité de négociation afin d'accéder au ressources des autres agents du système. Les trois classes d'agents du système UMDL sont (voir Figure 2.4) :

1 - UIAs (User Interface Agents) : ils fournissent le moyen de communiquer avec l'interface usager. Cet agent encapsule les requêtes en les adaptant au format du système UMDL. Il permet, entre autres, de créer un profil usager afin d'être utilisé par les agents médiateurs dans le processus de recherche.

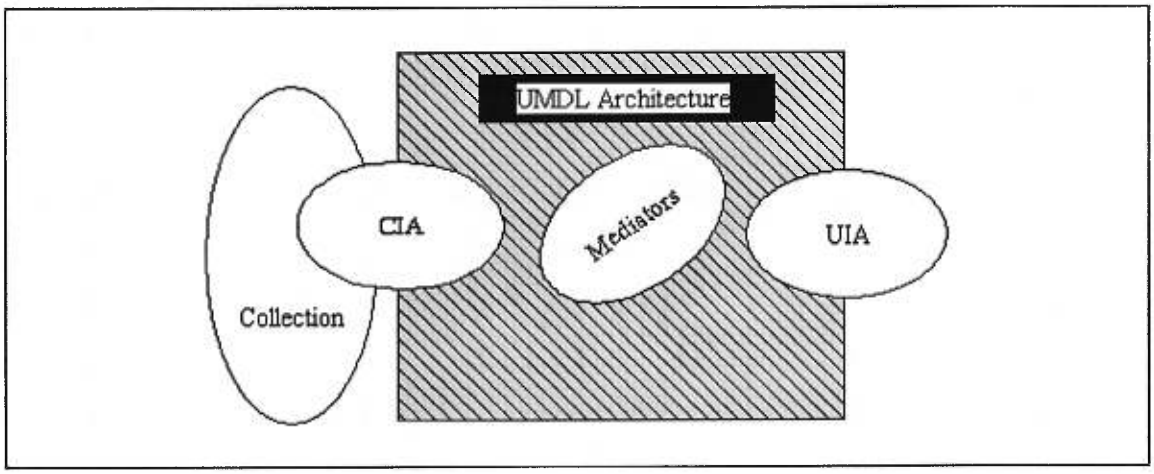


Figure 2.4 : L'architecture UMDL et les différentes classes d'agents qui la composent

2 - MAs (Mediator Agents) : ils remplissent une variété de fonctions comme: le suivi de l'évolution d'une requête, la prise en charge des requêtes à partir de l'interface usager, la transmission des résultats d'une recherche. Deux types d'agents sont actuellement utilisés dans le système. L'agent de planification (AP) qui reçoit une requête et la dirige vers les collections appropriées et l'agent d'enregistrement (AE) qui permet de capturer les adresses et le contenu des collections qui existent dans le système.

3 - CIAs (Collection Interface Agents) : ils permettent l'accès au contenu d'une collection. Comme exemple, un CIA pourrait utiliser le standard Z39.50 pour interroger la collection.

A titre d'exemple, le traitement d'une requête de recherche par auteur fera intervenir de la manière suivante les différents agent décrits précédemment: une fois la requête

soumise à l'UIA, celui-ci contacte le AE afin de déterminer le planificateur de requêtes pouvant faire des recherches par auteur. Une fois le planificateur identifié, celui-ci contact le AE et récupère les adresses des collections à consulter. Celles-ci sont accédées via CIA.

Les points les plus importants qui ont guidé le choix des chercheurs sont la nécessité d'une architecture configurable à volonté et capable de s'adapter à une rapide évolution des besoins. Ces besoins se manifestent tant sur le plan de la variété des services à offrir que sur le plan de la diversité des collections accessibles en ligne par les usagers.

L'utilisation d'agents jouissant d'autonomie, combinée à un contrôle décentralisé a permis de remplir l'ensemble des objectifs que se sont fixés les concepteurs du système. Cependant, comme nous le verrons dans la section suivante, certains problèmes restent toujours posés.

2.4 Synthèse et Discussion

À la lumière de l'étude faite précédemment, nous pouvons remarquer que les aspects les plus importants abordés dans les différents projets étaient en rapport avec les points suivants :

- interopérabilité entre systèmes,
- extensibilité,
- expansion,
- communication et standards adoptés,
- organisation des services,
- gestion de catalogues de données.

Les architectures qui ont attiré notre attention sont les systèmes UMDL et Harvest. Les caractéristiques principales que présentent ces deux types de systèmes sont les suivantes:

- utilisation de services distribués, ce qui permet l'extensibilité et l'expansion d'un système,
- fonctionnement en mode collaboratif, un mode qui permet une interaction entre les composants pour une éventuelle aide dans le processus de recherche; en particulier, dans UMDL, les composantes du système sont implantées comme des agents dotés d'une certaine autonomie,
- utilisation d'objets cache pour la restitution des résultats d'une recherche.

Les lacunes auxquelles nous devons faire face sont les suivantes:

a) Les niveaux d'indexation : nous remarquerons qu'il y a un seul niveau d'indexation créé pour les documents dans les systèmes étudiés. Ainsi, un courtier n'a pas de vue globale des collections de documents, d'où la difficulté à gérer de grandes quantités de données. L'une des spécifications dans notre projet est d'utiliser plusieurs niveaux d'indexation, ceci sera décrit en détail dans le chapitre 3.

b) Les agents de UMDL sont tous situés au même niveau : chaque agent effectue des tâches similaires. Bien souvent, pour effectuer une tâche complexe comme la recherche, un agent a besoin d'agents plus simples pour effectuer des sous-tâches. Ainsi, nous allons proposer l'utilisation d'un système d'agents organisés en plusieurs niveaux afin de répondre à des tâches de difficultés variées. Par exemple, nous pouvons utiliser un agent pour gérer à un niveau local une base de données et à un niveau supérieur un autre agent pour s'occuper des collaborations avec d'autres agents.

c) L'utilisation d'un standard pour l'interrogation : ce standard permettra une plus grande interopérabilité entre les différentes bibliothèques existantes et d'avoir un plus grand champ de recherche. Dans notre cas, nous avons exploré la possibilité d'utiliser le standard Z39.50.

d) Identification des informations permettant de décrire une collection comme par exemple la langue des documents de la collection, le type de documents (image, bande sonore,...).
e) Utilisation de services variés comme celui de l'expansion de requêtes ou celui de la traduction de requêtes pour les recherches multilingues et celui de la présentation des résultats sous différents formats.

L'étude des architectures existantes a fait ressortir la nécessité d'établir un système qui serait une combinaison des architectures proposées dans les systèmes UMDL et Harvest. Une architecture fortement intégrée, similaire à celle de Harvest impliquerait un traitement centralisé, ce qui rend cette solution inintéressante vu la lourdeur et la vulnérabilité qui en découlent. Dans le cas d'une architecture similaire à celle de UMDL, nous avons un traitement distribué. Cependant, cette dernière est difficile à gérer car un agent doit effectuer beaucoup de tâches à différents niveaux.

L'architecture proposée dans notre étude utilise certains des concepts introduits dans Harvest et UMDL, mais nous proposons une hiérarchie parmi les agents de recherche dans le but de rendre le processus de recherche plus performant. Nous décrivons cette architecture en détail dans le chapitre 3.

2.5 Notion de Méta-informations

Afin de faciliter l'interrogation des documents multimédia, il est nécessaire de définir les métadonnées qui permettront la description de ces documents. Nous en faisons un bref survol dans cette section.

Par méta-information, nous désignons des informations sur les informations. Ces informations décrivent des aspects différents sur les documents qui vont permettre une recherche efficace.

Nous avons plusieurs types de documents à répertorier parmi lesquels nous pouvons citer des documents textuels, comme les livres, et des documents images. Ces deux types de documents vont se baser sur différentes méta-informations pour être décrits.

Les documents textuels utiliseront des méta-informations classiques comme un nom d'auteur, une date de parution, la langue de rédaction, etc. En ce qui a trait aux images, deux types de méta-informations sont à distinguer, celles qui dépendent du contenu et celles qui sont indépendantes du contenu. Dans le premier cas, ce sont des informations basées sur le contenu du document ou associées par l'usager. Dans le second cas, elles correspondent aux méta-informations sur les droits et propriétés, sur le stockage et sur la représentation des documents visuels. Parmi les informations dépendantes du contenu, nous distinguons:

***Les annotations textuelles*:** ce sont des informations relatives au contenu et au contexte de l'image. Ces méta-informations peuvent être des mots-clé et/ou des descriptions sur le contenu de l'image ou sur le contexte dans laquelle elle a été prise. Comme exemple, nous pouvons avoir pour l'image ci-dessous l'annotation textuelle : mur, briques rouges.

Les méta-informations extraites sont des informations sur les caractéristiques visuelles des images, elles sont générées de manière automatique par des techniques de traitement d'images. Ces techniques permettent de capturer les propriétés visuelles des images sous la forme de données structurées (numériques ou textuelles) qui sont ensuite utilisées pour constituer les structures d'index. Les caractéristiques visuelles généralement utilisées pour décrire les documents visuels sont les suivantes :

Les moyennes de couleurs: calculées dans un modèle ou espace de couleurs (RGB, par exemple), elles déterminent la quantité de chaque couleur contenue dans des pixels et caractérisent une image en fonction de la moyenne des couleurs principales qui la composent. Les moyennes de couleurs sont très utiles pour rechercher des zones de couleurs plus ou moins uniformes.

Les histogrammes de couleurs: définis à partir des modèles de couleurs, ils fournissent une distribution de couleurs de plusieurs dimensions. Chaque dimension correspond à la représentation d'une couleur de l'image de l'objet ou de la zone dans un modèle de couleurs donné. La valeur de la dimension de l'histogramme de couleurs dépend du nombre de couleurs nécessaires pour capturer la distribution des couleurs dans une image. En général, ce nombre varie entre 64 et 256; contrairement aux moyennes de couleurs, les histogrammes de couleurs sont invariants en translation et en rotation [Swain, 91].

La texture: elle permet d'identifier des objets ou des zones à partir de la structure de leur surface et de déterminer leur localisation dans une image à partir des différences de structures spatiales. La texture est très utile pour décrire les éléments - comme le gazon, le sable, le ciel, une foule de personnes, des immeubles, etc.- qui n'ont pas de contour ou de forme spécifique et qui sont mieux décrits par des propriétés collectives.

La forme: la forme d'un objet ou d'une région peut être décrite par des caractéristiques globales telles que le périmètre, la hauteur, la largeur, le diamètre, la circonférence, la surface, la circularité, l'excentricité, l'orientation de l'axe de gravité ou par des caractéristiques locales comme l'échelle, l'orientation et les déformations subies par un objet ainsi que son angle d'orientation qui dépendent du type de forme et du contexte de l'application.

Les relations spatiales: les caractéristiques spatiales décrivent les relations de direction, les relations topologiques et les relations de distance. Les relations de direction sont les relations faisant appel à une direction selon un axe (nord, nord-ouest). Les relations

topologiques sont les relations de voisinage, elles combinent la frontière (rencontre en), l'intérieur (contient, couvre, à l'intérieur de) et l'extérieur (séparé de) des objets et/ou des zones. Les relations de distance décrivent l'espacement entre les objets (près, loin) [Yapo, 98].

L'utilisation de standards dans la conception et l'implantation de méta-informations est un aspect important qui permet d'assurer la cohérence des informations échangées entre les bibliothèques numériques. Ces standards fournissent les règles sur l'organisation, la syntaxe et les formats d'échanges des données. Une introduction aux standards les plus connus dans le domaine des bibliothèques numériques est présentée dans la section qui suit.

Caractéristiques des standards de méta-informations

Les standards utilisés dans le développement de systèmes de gestion de documents visuels sont principalement des standards qui permettent de normaliser la structure des données, le contenu des données, la valeur des données et leur communication. Parmi ces standards nous pouvons citer : le *Dublin Core [Dublin core]*, le *Content Standard for Digital Geospatial Metadata* et le *Core Categories for Visual Resources* [Yapo, 98].

Ces standards présentent un certain nombre de critères pouvant être utilisés pour une éventuelle comparaison. Ces critères sont brièvement décrits ci-dessous [Yapo, 98] :

1. *l'intégration sémantique:* ce critère permet d'homogénéiser la sémantique des méta-informations de façon à ce que ces dernières puissent décrire de façon cohérente et consistante un grand nombre de documents visuels;
2. l'*extensibilité:* ce critère rend possible l'ajout et l'utilisation de méta-informations non définies par le standard;
3. la *généricité:* ce critère qui sert à déduire les méta-informations spécifiques aux applications modélisées à partir des méta-informations du standard sans altérer leur format de base;

4. la *flexibilité:* ce critère simplifie la spécialisation des méta-informations et aide à obtenir des modèles conformes aux spécificités des domaines d'application modélisés;
5. les *relations avec d'autres standards*: ce critère accroît l'adaptabilité des modèles et la cohérence des informations échangées par les systèmes;
6. *la simplicité de création et de maintenance*: le respect de ce critère accroît l'utilisation des standards car leur mise en œuvre ne nécessite pas les compétences d'un expert en catalogage pour la création et la maintenance des méta-informations;
7. les *types de méta-informations utilisés*: on distingue les *méta-informations dépendantes du contenu* et les *méta-informations indépendantes du contenu.*

ANSI/NISO sont des organismes de standardisation et de spécification de protocoles qui ont proposé un standard pour les applications en recherche documentaire basé sur des méta-données descriptives : le standard Z39.50 [ZIG 97]. Nous en faisons une brève description dans la section suivante.

2.6 Présentation du Standard Z39.50

A l'origine, en 1984, le standard avait été proposé pour des documents bibliographiques. Dès 1990, un groupe (ZIG[2]) a vu le jour; ce dernier regroupait des fabricants, des vendeurs, des consultants, des fournisseurs d'informations et des universités dont le souhait était de fournir ou d'accéder à différents types d'informations via le standard Z39.50. Les informations accédées sont variées: les notices bibliographiques, du texte, des images, des données financières, des informations d'utilité publique et des données des domaines de la spécialisation scientifique (chimie).

La puissance du standard Z39.50 tient dans le fait qu'il sépare l'interface d'interrogation des serveurs d'informations. Il fournit une vue consistante d'une large variété de sources d'informations et une implémentation plus libre des clients [Corey 94].

---

[2] Z39.50 Implementation group

Z39.50 a l'avantage d'uniformiser la recherche d'information, et si pour le moment ce protocole est appliqué essentiellement par des professionnels de l'information bibliographique, et plus particulièrement par les bibliothécaires qui en sont les protagonistes, certains pensent sérieusement que le couple WWW / Z39.50, formé de deux entités naturellement appelées à se compléter, devrait dépasser le cadre de l'information bibliographique pour s'appliquer à tous les besoins en recherche d'information.

Les Eléments de Base du Standard

Z39.50 est un protocole orienté session, c'est-à-dire qu'une session persistante est démarrée lorsqu'une connexion au serveur Z39.50 est effectuée. Dans ce cas, une connexion est maintenue tant que la session reste ouverte [Corey 94]

Les protocoles orientés session sont plus performants que les protocoles orientés transaction qui demandent qu'une connexion soit établie pour chaque envoi de message, sans négliger le fait que c'est le moyen qui permet de faire le raffinement itératif du résultat de recherche. C'est aussi le moyen qui permet au serveur et au client de négocier les services à maintenir tout au long de la session. Ce n'est pas le cas du protocole HTTP pour lequel les messages échangés doivent contenir la description de certaines caractéristiques que le serveur doit avoir et ceci pour chaque transaction.

Dans sa forme la plus simple, le standard Z39.50 est un protocole synchrone. Dans ce cas, le client envoie un message au serveur et attend la réponse .

Le protocole spécifie les formats et les règles gouvernant les échanges entre un client et un serveur. Ces échanges permettent au client de demander au serveur de chercher des documents qui vérifient des critères spécifiques. Z39.50 définit un ensemble d'attributs standards pour identifier et rechercher des documents. Par exemple, la requête qui permettra de rechercher tous les documents dont l'auteur est "Bowman" et dont le titre contient les mots "bibliothèques numériques" est la suivante:

And (Bowman[1,1003],"Bibliothèques numériques"[1,3])

Ce protocole permet l'interrogation simultanée de banques de données par l'envoi d'une requête unique à *plusieurs serveurs distants* sans se préoccuper de l'interface propre au module de recherche utilisé par les usagers directs de ces banques de données.

Le standard Z39.50 et les *services* qu'il fournit prévoient non seulement la navigation dans les bases de données selon le mode hypertexte, mais également le parcours de listes d'indexes sur un serveur. Il est implémenté au-dessus du protocole TCP/IP et permet à tout client Z39.50 d'envoyer une requête de recherche indiquant une ou plusieurs bases de données et les paramètres qui permettent de préciser si les réponses doivent inclure ou non les enregistrements trouvés dans la réponse. La responsabilité de la récupération des enregistrements est laissée à la charge du client. En effet, le serveur organise les enregistrements trouvés en ensembles et ainsi il permet au client de récupérer ceux qui l'intéressent grâce à leurs positions. Les possibilités de base qui doivent être offertes sont les suivantes :

- Le client doit pouvoir indiquer un ensemble de paramètres lui permettant de préciser les données de l'enregistrement à récupérer. Par exemple, le client peut demander au serveur de retourner l'ensemble des enregistrements trouvés si leur nombre ne dépasse pas cinq; sinon, une description sommaire de chaque enregistrement est récupérée,
- Le client peut indiquer un format à respecter pour les réponses,
- Le client peut nommer l'ensemble des enregistrements retrouvés pour les référencer dans des transactions ultérieures,
- Il peut détruire cet ensemble d'enregistrements,
- Le serveur peut imposer un accès réduit en demandant une authentification avant toute opération de recherche.

CNIDR (Clearingohouse for Networked Information Discovery and Retreival) est l'un des organismes qui a proposé une implémentation du standard Z39.50. Le système,

distribué gratuitement, s'appelle Isite. C'est un système d'information Internet complet qui peut intégrer plusieurs systèmes de gestion de base de données et différents protocoles ouverts comme le WWW et la messagerie électronique. La version 1.04 d'Isite inclue les éléments suivants:

- Toutes les applications de communication Z39.50 (client et serveur),
- Une API de recherche,
- Un système d'indexation et de recherche de texte,
- Une passerelle http / Z39.50.

Nous allons choisir le standard Z39.50 principalement pour les raisons suivantes:

- Sa facilite d'utilisation comparé à d'autres standards comme *Dublin core* qui lui reste très abstrait,
- Il répond à nos besoins essentiels de recherche de documents textuels. Cependant, une extension sera nécessaire pour la recherche d'images selon des attributs visuels,
- Il existe un ensemble d'outils qui nous permettent d'utiliser le standard de manière simple et efficace.

Cependant, il faudra l'étendre au cas de la recherche des images, point que nous avons abordé dans le chapitre quatre.

Chapitre 3

3 Architecture Multi-Agents à Extension de Services

- Conception -

Comme il a été mentionné dans le chapitre précédent, il existe plusieurs types de systèmes de bibliothèques numériques en utilisation ou au stade expérimental. Nous avons mis en évidence les limites de certaines architectures.

Les systèmes qui ont retenu notre attention sont particulièrement les systèmes UMDL et Harvest. Dans notre cas, la problématique est de concevoir un système qui permet de gérer des documents multimédia, c'est-à-dire des documents constitués d'images et/ou de textes. Parmi les caractéristiques que doit présenter le système nous pouvons citer:

1. Prendre en charge de grandes capacités de données,
2. Faciliter l'extension du système,
3. Permettre l'expansion du système,
4. Permettre l'interopérabilité avec d'autres systèmes,
5. Fournir des services variés tels que l'interrogation multilingue.

Des décisions de conception ont été prises pour mettre au point l'architecture du système. En effet, nous avons fait en sorte que les caractéristiques citées précédemment soient assurées. Pour ce faire, nous avons:

- défini une organisation hiérarchique des catalogues de données,
- utilisé un système multi-agents permettant de fournir plusieurs services,
- choisi le standard Z39.50 pour l'interrogation des données.

L'organisation multi-niveaux des catalogues permet de gérer une plus grande quantité de données et d'accélérer le processus de recherche. Nous distinguons deux niveaux d'agents de recherche. Le premier niveau permet d'identifier les sites susceptibles de contenir les documents recherchés. Le second niveau effectue une recherche plus en profondeur.

L'architecture multi-agents est basée sur plusieurs entités permettant une gestion distribuée. Cette distribution des responsabilités permet d'augmenter la fiabilité, la robustesse et la performance de tout le système. Elle permet aussi de réduire les coûts en termes de communications dans le réseau et de répartir les traitements.

D'après Jennings [Jennings 96], un agent est un programme capable de prendre ses propres décisions, en se basant sur la perception de son environnement dans l'objectif d'atteindre un ou plusieurs buts. Jennings classe les agents en trois niveaux. Au niveau le plus simple, il y a des agents « gophers » qui exécutent des tâches simples basées sur des règles prédéfinis, comme par exemple informer le gestionnaire quand le stock atteint un seuil donné. À un niveau plus haut, on trouve les agents fournisseurs de services. Ces agents réalisent des tâches bien définies à la demande de l'utilisateur, comme par exemple chercher le billet d'avion le moins cher pour une destination donnée. Et enfin, les agents du niveau supérieur qui proposent des informations ou des services à l'utilisateur, sans être explicitement sollicités. C'est le cas par exemple d'un agent qui

prévient l'utilisateur qu'il y a des réductions sur des séjours touristiques, sachant que l'utilisateur est intéressé par les voyages.

Cette classification répond bien aux responsabilités assignées aux agents du système que nous avons conçu. Ces agents se caractérisent par leur simplicité et se situent entre ceux du système Harvest, qui sont statiques, et ceux du système UMDL, qui sont dynamiques et plus complexes.

Les caractéristiques avancées par Wooldridge dans la définition faible d'un agent [Wooldridge 94] sont en partie vérifiées pour nos agents.
Parmi ces caractéristiques, nous avons:

- L'autonomie: les agents opèrent sans intervention directe des humains; ils ont un certain contrôle de leurs actions.
- La réactivité: un agent perçoit son environnement et réagit dans les temps aux changements dans l'environnement. C'est le cas pour la mise à jour de la liste des sites actifs et ceux qui sont hors service.

Pourquoi ces deux niveaux d'agents?

Si un seul niveau d'agents est utilisé pour une requête, tous les agents sont contactés. Mais plusieurs parmi eux ne sont pas pertinents pour l'exécution de la requête. Par exemple, il est inutile de contacter un agent qui gère une base de documents en chimie pour une requête en informatique. De même, un agent qui gère des documents en français ne sera pas utile pour une requête qui cherche des documents en anglais.

Un second problème est lié à la fusion des résultats obtenus de différents agents. Dans la plupart des cas, un agent va répondre à n'importe quel type de requête par un ensemble de documents. Mais la qualité des réponses provenant des agents est différente. Si on demande à tous les agents de fournir un ensemble de documents, la fusion sera très

difficile car il est souvent ardu de distinguer des documents provenant des sources pertinentes de ceux provenant de sources non pertinentes.

Ainsi, nous prévoyons une première recherche des agents selon les caractéristiques de la requête. Une seconde recherche plus détaillée sera exécutée par les agents identifiés. L'utilisation de plusieurs classes d'agents est une solution qui facilitera l'extension du système tout en décentralisant les traitements et par la même occasion répondre à des besoins d'expansion par l'introduction de nouvelles bases d'information.

Le troisième aspect important que nous avons pris en compte concerne l'interopérabilité avec d'autres systèmes. Une façon de faire cela est d'utiliser un standard d'interrogation comme le standard Z39.50.

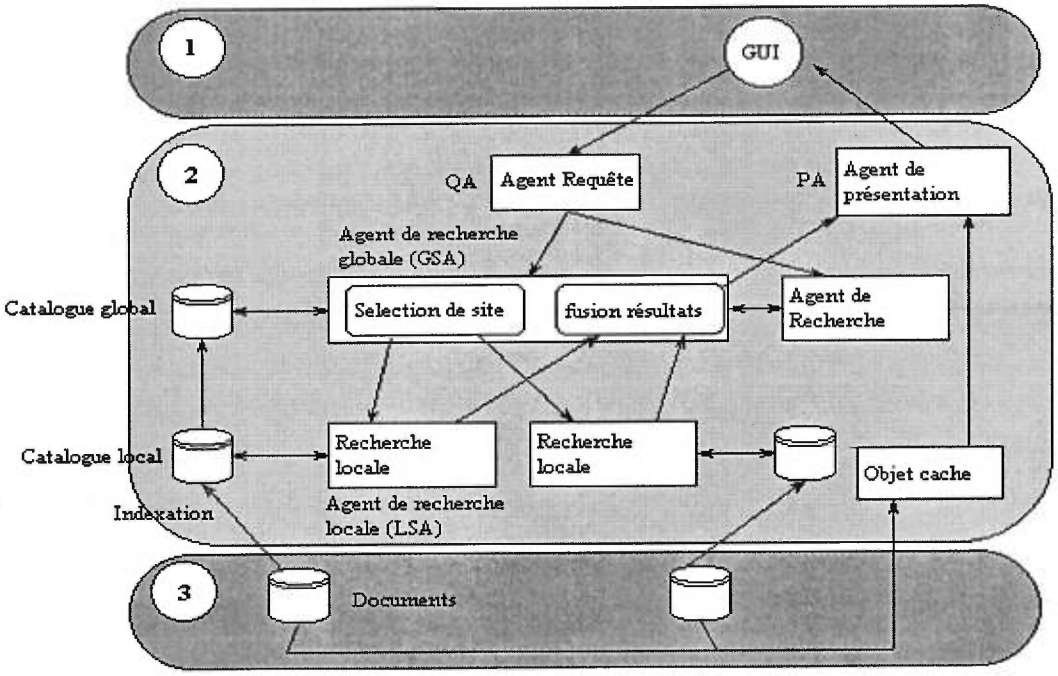


Figure 3.1 : Architecture globale du système. Trois parties importantes, l'interface utilisateur (1), le système multi-agents (2) et le système de stockage (3).

La Figure 3.1 donne un aperçu de l'architecture globale du système. Nous pouvons y remarquer les différents types d'agents impliqués. Une requête usager est transmise à un

agent de requête dès sa soumission au système. Après analyse, elle est dirigée vers l'agent de recherche global qui gère un catalogue global des sites répertoriés dans le système. Le but de cet agent est d'identifier les sites pertinents à interroger. Une fois ces sites identifiés, la requête usager est transmise aux différents agents de recherche locale qui gèrent à leur niveau un catalogue contenant l'ensemble des informations permettant de retrouver les documents recherchés. Ces agents vont faire parvenir les résultats d'une recherche à l'agent de recherche global qui les a sollicité afin d'être fusionnés et renvoyés à l'usager via un agent de présentation. Un mécanisme de cache est utilisé afin de permettre à l'usager d'accéder aux documents trouvés sans solliciter inutilement le système.

Une première décomposition de cette architecture nous amène aux trois composantes principales comme illustrés dans la Figure 3.2.

- L'interface graphique au système,
- Le système de traitement des requêtes,
- Le système de stockage.

Dans les sections suivantes, nous allons décrire les trois composantes dans le détail.

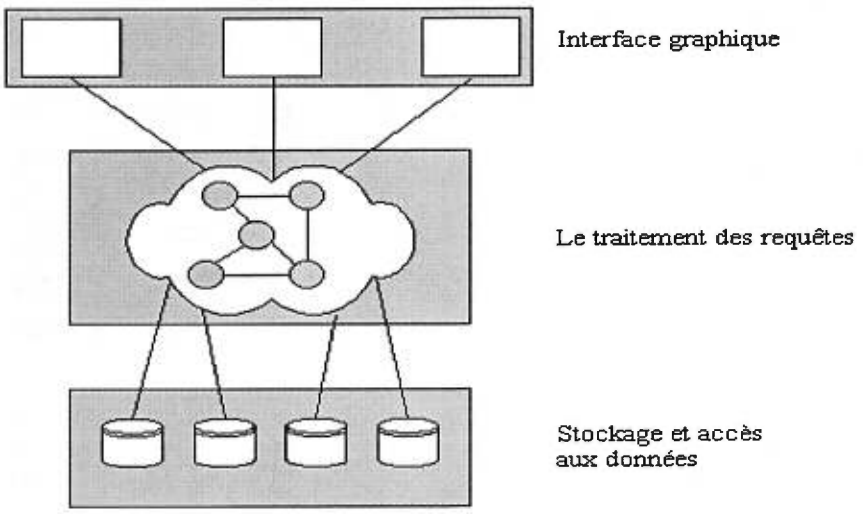


Figure 3.2 : Architecture à trois niveaux. La partie interface graphique, la partie traitement et la partie stockage.

3.1 Le Système de Traitement de Requêtes

3.1.1 Aperçu général du système

Du point de vue logique, le système est organisé en une architecture hiérarchique. La Figure 3.3 montre les composantes principales. Nous y observons les agents suivants:

L'agent superviseur (SA): la gestion et la coordination des agents qui composent le système doivent être assumées par un agent unique, le *superviseur*. C'est par lui que se font connaître les autres agents. Il dispose d'un certain nombre de services qui vont assurer le bon fonctionnement du système, tels que l'enregistrement, le retrait et la consultation.

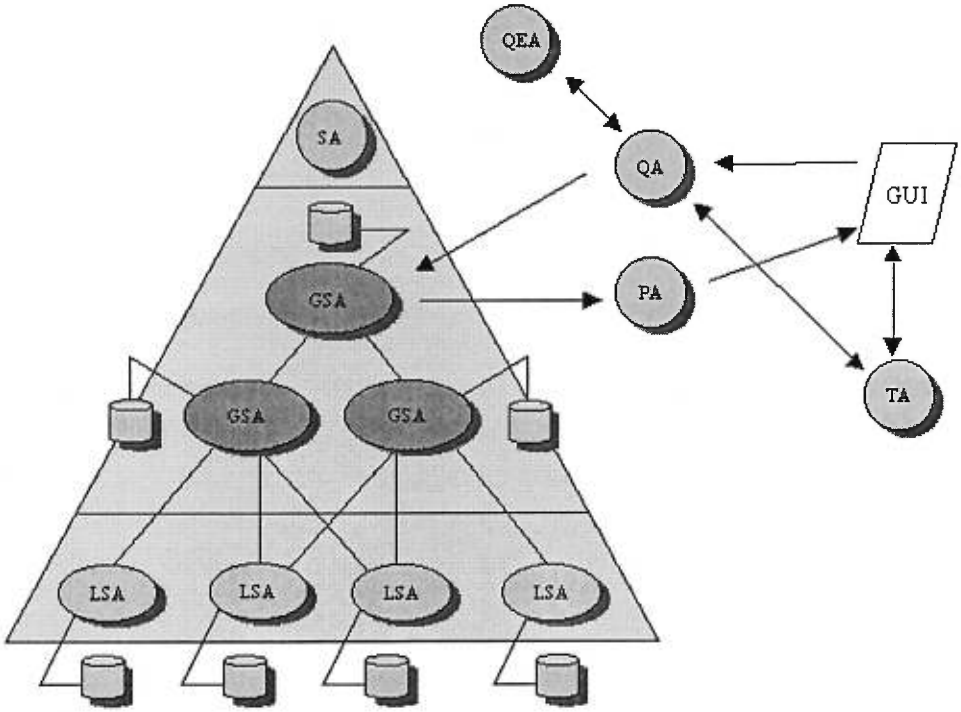


Figure 3.3 : Organisation pyramidale du système. Les GSA sont au niveau le plus élevé, puis à un niveau inférieur viennent les LSA.

Les agents de recherche globale (GSA) : plusieurs entités de ce type peuvent exister dans le système. Ils gèrent une base de données de serveurs, appelée catalogue global,

qui supportent les bases de données qui renferment les documents. Un GSA peut prendre à sa charge un ou plusieurs LSA.

Les agents de recherche locale (LSA) : un agent LSA peut prendre en charge une ou plusieurs bases de documents. Ils reçoivent les requêtes des agents (GSA), les traitent et retournent les résultats.

Ces trois types d'agents sont établis en permanence dans le système alors que d'autres agents seront créés dynamiquement.

L'agent de requête (QA) : cet agent est responsable des requêtes soumises par l'usager. Dans certains cas et selon les exigences de l'usager, cet agent fera appel aux services d'autres agents qui existent dans le système. Par exemple, un agent peut offrir un service permettant d'étendre une requête usager en se basant sur un thésaurus. Une fois la requête traitée, elle est envoyée aux agents GSA.

L'agent de traduction (TA) : cet agent offre un service de traduction de texte d'une langue dans une autre et est utilisé directement par l'usager à travers l'interface graphique.

L'agent de présentation (PA) : cet agent permet de formater les documents selon les exigences de l'usager. Par exemple, si l'usager veut voir les documents sous le format HTML, alors les services de ce type d'agents seront utilisés avant l'affichage des documents à l'usager.

L'agent d'expansion de requêtes (QEA) : cet agent permet l'expansion de requêtes afin d'élargir le champ de recherche.

En plus de ces agents, il existe un objet cache qui va permettre de mémoriser temporairement les résultats d'un usager lui permettant ainsi de retrouver les derniers documents consultés.

Les agents des trois dernières catégories sont créés dynamiquement à la demande d'un usager et en fonction de ses besoins. Ainsi, nous évitons de surcharger le système par des services qui risquent de ne pas être utilisés.

Dans les sections suivantes, on identifie les agents utilisés, leurs caractéristiques ainsi que leur fonctionnement et toutes les interactions nécessaires pour remplir leur mission.

3.1.2 Description des agents du système

L'agent superviseur (SA)

Cet agent est unique et est le premier à être mis en service dans le système. Après sa création et son initialisation, il enregistre les informations nécessaires à son identification dans le réseau dans un fichier de configuration *SACF (Supervisor Agent Configuration File)* indiquant l'adresse réseau de la machine sur laquelle il a été créé et le numéro de port sur lequel il se met à l'écoute des agents clients. Ce fichier de configuration est le lien entre les autres agents du réseau et le *SA*. Tous les agents accèdent à ce fichier à travers un serveur WEB.

Plusieurs tâches sont assignées au SA. En effet, il a comme première tâche la gestion de l'intégrité du système, c'est à dire de vérifier que les agents inscrits sont effectivement actifs. Dans ce cas, toute anomalie de dysfonctionnement détectée implique le retrait temporaire ou permanent de l'agent hors service.

La deuxième tâche qui lui incombe concerne la répartition de la charge que les agents ont à absorber. Le *SA* peut être amené à orienter les requêtes en fonction de la charge de chacun des agents impliqués dans les recherches de documents. Il est clair que l'information de charge est primordiale au *SA* dans le processus de décision. Donc, parmi les informations associées à un agent qui s'enregistre, nous avons sa charge courante. Celle-ci sera mise à jour régulièrement.

La gestion des agents du système se fait grâce à un certain nombre de listes constituées par catégorie d'agent. Tous les agents qui offrent leurs services s'enregistrent auprès du

superviseur afin de se faire connaître et de permettre au *SA* de remplir ses fonctions de contrôleur. Ces listes sont accessibles par l'intermédiaire de services d'enregistrement et de lecture. Le service d'enregistrement permet d'enregistrer un agent dans la liste qui lui correspond et le service de lecture permet à un agent de lire la liste d'une catégorie d'agents. Dans notre système, nous avons créé plusieurs types de listes et qui sont: la liste GSAL (Global Search Agent List) pour les agents GSA, la liste QEAL (Query Expanding Agent List) pour les agents d'expansion de requêtes, la liste QTAL (Query Translation Agent List) pour les agents qui font de la traduction de requêtes (voir Figure 3.4).

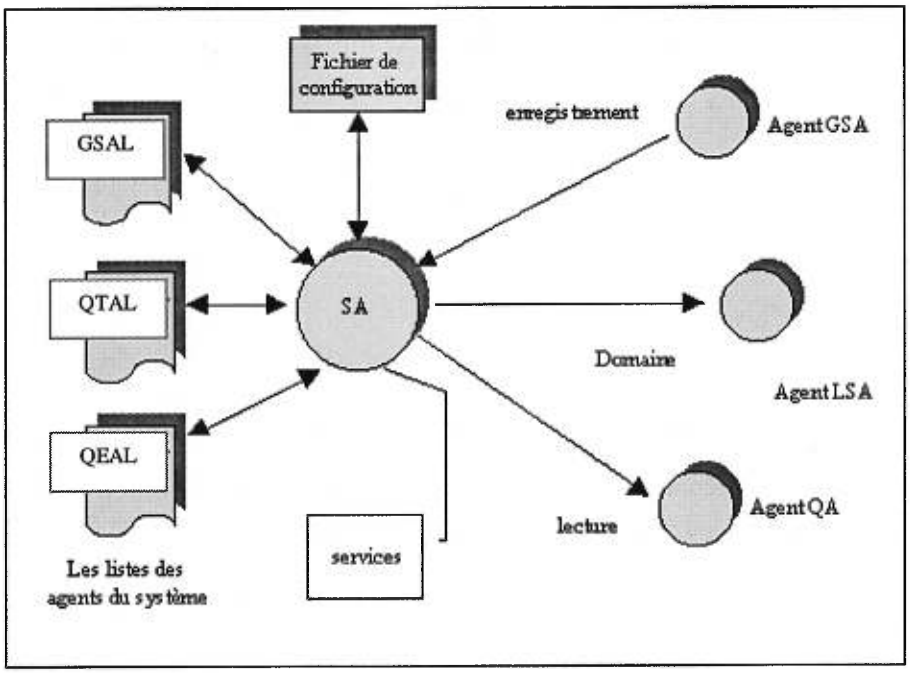


Figure 3.4 : Description de l'agent SA. Les agents LSA, GSA et QA utilisent les services offert par le SA. Différentes listes sont gérées par l'agent superviseur.

A) Description de la liste GSAL

Les agents GSA sont organisés en domaines (réseau). Dans chaque domaine, les agents GSA se voient attribuer un nombre plus ou moins équitable d'agents LSA à gérer. Ainsi, il sera facile d'équilibrer la charge de travail entre les GSA du système.

La structure de la liste est la suivante:

Domaine 1 :

GSA1 :

Adresse,
Nombre de LSA gérés,
Charge actuelle,
État actuel.

GSA2 :

Adresse,
Nombre de LSA gérés,
Charge actuelle
État actuel

B) Description des listes QTAL et QEAL

Ces deux listes contiennent les références des agents qui offrent respectivement les services de traduction et d'expansion de requêtes. Dans la première liste sont répertoriés tous les agents ainsi que les différents types de traduction. La structure de la liste est la suivante:

<Référence agent >
<langue A, langue B>
<langue A, langue C>
<langue A, langue D>
...

Dans la seconde liste sont regroupés les agents qui permettent de faire l'expansion des requêtes soumises. L'expansion permet de composer une requête plus complète en utilisant un thésaurus dans un domaine précis. Pour chaque agent est indiqué le ou les domaines pour lesquels il est spécialisé.

<Référence agent >
<Domaine 1>
<Domaine 2>
...

C) Description des services offerts par SA

Enregistrement d'un agent GSA

Ce service permet à SA de déclarer tout agent GSA nouvellement mis en service dans une liste appelée GSAL. La déclaration permettra l'utilisation de l'agent dans le processus de traitement des requêtes utilisateurs.

La liste GSAL contiendra toutes les informations nécessaires à la gestion de l'ensemble des agents GSA du système comme par exemple l'état de fonctionnement de l'agent, l'adresse de l'agent et la charge actuelle. Cette dernière information est calculée en fonction du nombre de requêtes en cours de traitement par l'agent.

Le système est capable de prendre en compte les nouvelles bases de données introduites. Pour ce faire, les agents LSA utilisent eux aussi un service d'enregistrement à travers lequel ils se déclarent en envoyant leurs références. En réponse, le SA lui renvoie les références de l'agent GSA qui les prendra en charge. Basé sur les références de l'agent GSA, l'agent LSA fait parvenir toutes les données qui le décrivent ainsi que les informations des bases de données qu'il gère.

Un agent GSA peut à tout moment se retirer du système en utilisant un service. Le retrait peut être temporaire ou permanent. Dans le premier cas, le SA utilise une information pour indiquer que l'agent en question est momentanément hors service. Dans le second cas, le SA retire l'agent de sa liste et réorganise les agents LSA qu'il gérait.

Lecture de la liste GSAL par les QA

Les agents QA utilisent ce service afin de lire la liste de GSA qui se trouve en service dans le système. A la réception de ces informations, l'agent QA pourra envoyer sa requête aux différents agents GSA.

Identification du domaine d'un agent

L'agent SA offre un service permettant ultérieurement aux agents LSA de déterminer à quel domaine ils appartiennent. Ce service va permettre à un agent de savoir auprès de quel agent GSA s'enregistrer.

L'agent de requête (QA)

C'est l'agent qui est directement en relation avec l'interface graphique utilisée par l'usager pour transmettre des requêtes de recherche. Une fois la requête composée, un objet requête (OBR) est créé. Cet objet contient les options de la recherche (par exemple domaine de spécialisation, langue, type de document) et la requête proprement dite. Une fois l'OBR soumis à QA, il est envoyé aux différents agents GSA contenus dans la liste GSAL renvoyée par SA.

Cet agent peut être amené à utiliser les services d'autres agents avant de retransmettre l'OBR aux GSA; c'est le cas lorsque l'usager demande une extension de requête. L'agent QA extrait de l'objet ORB la partie requête et l'envoie à l'agent qui possède le service. Comme pour la liste GSAL, le QA demandera au SA la référence d'un agent pour pouvoir utiliser ses services. Le SA retourne à la demande les références d'un agent en service sinon il lui indique qu'aucun agent n'est disponible.

Un agent de type QA est crée à chaque envoi d'une requête usager. Il est détruit après l'étape de création de l'objet OBR envoyé aux différents GSA.

L'agent de recherche globale GSA

Le système peut avoir plusieurs GSA. Chacun d'eux prend en charge un agent de type LSA utilisé pour la gestion d'une base de données.

Comme nous l'avons mentionné au tout début de la description du système, ces agents se situent au dessus des LSA. Il gèrent chacun un catalogue global contenant les informations des sites gérés par ce GSA.

En se basant sur les informations contenues dans l'objet requête OBR envoyé par l'agent QA, l'agent GSA détermine, après consultation de son catalogue global, les agents LSA à contacter pour traiter la requête de l'usager. Les informations disponibles dans le catalogue pour gérer un LSA sont les suivantes:

Catalogue global:

<Domaine de spécialisation> <site de recherche>

<langue> <site de recherche>

<type de document> <site de recherche>

Les champs ci-dessus permettent une classification des sites où doit se faire la recherche. Ainsi, si une requête fait référence à des sites dans le domaine de l'art, alors seuls les sites qui ont rapport avec le domaine de l'art seront sélectionnés pour une interrogation. Ces informations sont communiquées par les agents LSA aux agents GSA lors de leur déclaration dans le système.

Les domaines de spécialisation peuvent être hiérarchisés de façon à accélérer le processus de recherche. Par exemple, nous pouvons avoir un domaine scientifique dans lequel sont répertoriés différents domaines des sciences comme l'informatique, la chimie, etc. Le domaine de l'informatique est lui même composé de différents domaines comme les bases de données, l'intelligence artificielle, réseaux et télécommunication, etc. Lors de la soumission d'une requête est indiqué un domaine précis (base de données par exemple) et le système peut identifier le domaine principal (science) afin de retrouver les sites à contacter.

Parmi les services offerts par les agents GSA nous avons:

Le service d'enregistrement d'un agent LSA

Les LSA utilisent ce service après avoir déterminé à quel domaine ils appartiennent. Les informations envoyées lors de l'enregistrement concernent chacune des bases de données

gérées par LSA. Si le LSA est déjà enregistré alors il ne fait parvenir que les informations concernant les nouvelles bases de données introduites dans le système.

Le service de retrait d'un agent LSA

Un LSA a la possibilité de se retirer du système. Dans ce cas, il fait appel au service de retrait de l'agent GSA. Ce dernier retire de manière temporaire ou permanente l'agent qui en fait la demande. Si le retrait est définitif alors toutes les informations le concernant sont retirées du système et l'agent SA en est avisé.

L'agent GSA intervient à deux moments précis dans le traitement d'une requête:

1) Il s'implique à la réception de l'objet OBR envoyé par l'agent QA associé à l'interface usager et à la réception des résultats renvoyés par les agents LSA.

2) Une fois l'OBR reçu, l'agent GSA détermine un ou plusieurs sites où sont localisés les serveurs de données à contacter. Chaque serveur de données est géré par un agent LSA à qui sera transmise la requête Z39.50 générée à partir de l'objet OBR. A ce moment est créé un processus pour prendre en charge tout le processus de recherche locale. Ce sont les étapes 1, 2, et 3 de la Figure 3.5.

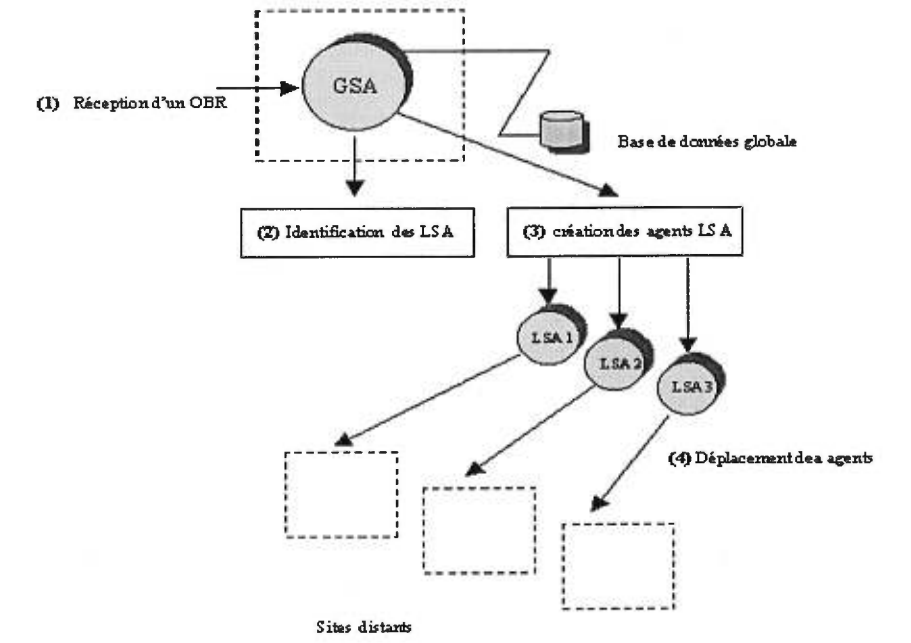


Figure 3.5 : Les étapes de traitement d'une requête

Une fois les LSA créés, le GSA se met en attente des résultats qui seront affichés par les LSA après la recherche. Les résultats des LSA et un résumé des documents trouvés sont triés et envoyés à un objet cache (OBCR) afin de les restituer à l'usager. Tous les LSA terminent leur exécution après avoir transmis les résultats au GSA. Ce processus est représenté par les étapes 4, 5, 6 et 7 de la Figure 3.6: recherche dans la base (4), retour de résultats (5), fusion et classification des résultats (6), transfert de la liste des résultats vers OBCR (7).

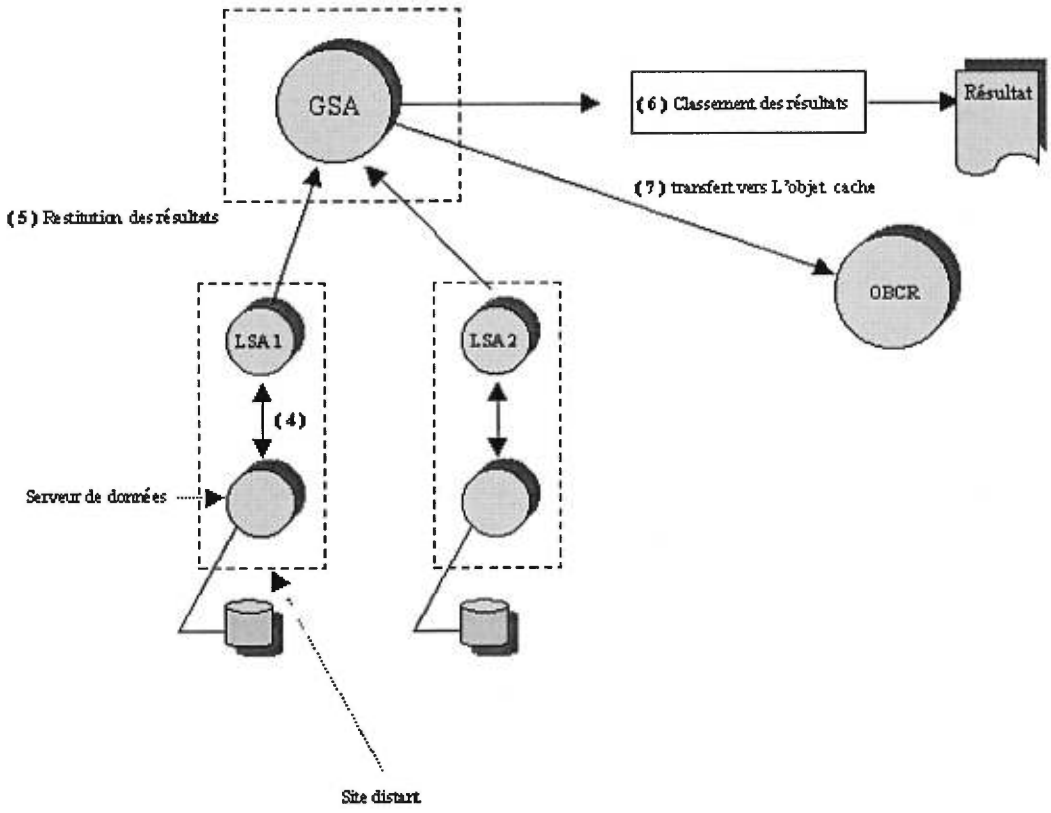


Figure 3.6 : Récupération des résultats par les LSA et restitution au GSA. Le GSA les transmet à l'objet cache après leur classement.

Pour fusionner en une seule liste les résultats retournés par les LSA, nous utilisons la technique qui se base sur la similarité normalisée [Shivakumar 95] des documents, c'est-à-dire, les similarités des résultats de chaque LSA sont normalisés entre 0 et 1. Les résultats des différents LSA peuvent ainsi être regroupés et classés.

L'agent de recherche locale LSA

Cet agent est utilisé pour envoyer la requête au serveur de base de données locale et transmettre les résultats à l'agent GSA.

La bande passante du réseau étant une ressource critique, il devient important de trouver des solutions efficaces pour éviter son gaspillage durant le processus d'exécution des requêtes. Dans notre cas, nous pouvons créer un client mobile qui se déplacera vers le

site distant afin de se connecter au serveur de données Z39.50 et lancer sa requête de recherche. Tous les échanges générés entre le serveur et l'objet mobile sont locaux et donc rapides. Les résultats obtenus sont traités localement et rapatriés vers l'agent GSA. Parmi les traitements effectués localement nous avons le choix préliminaire de certains documents à restituer et la compression des résultats, ce qui nous permet un gain de temps et nous évite de surcharger inutilement la réseau. Ainsi, bien que ce soit un choix d'implantation, nous avons montré dans ce chapitre cette migration de service LSA comme une propriété importante dans la conception d'un système de BN.

Nous avons indiqué dans la section qui décrit l'agent SA qu'il existait un service permettant de savoir auprès de quel agent GSA un agent LSA doit s'enregistrer. Une fois cette étape passée, le LSA contacte l'agent GSA et lui transmet toutes les informations qui le concerne pour être introduit dans la base de données globale. Cette opération est rendue possible grâce aux services offerts par les GSA.

Agent de traduction TA

Un agent de traduction est utilisé pour aider l'usager à composer ses requêtes dans une langue qui ne lui est pas familière. Cet agent est contacté directement à partir de l'interface usager lorsque celui-ci veut l'utiliser. C'est un serveur qui reçoit les requêtes sous une forme textuelle puis effectue éventuellement une traduction et affiche le résultat à l'interface usager afin d'être affiché. L'agent peut être simple ou sophistiqué. Dans le premier cas, il recevra des mots ou expressions à traduire et restituera la traduction. Dans le deuxième cas, il pourrait assister l'usager dans le processus de traduction de manière interactive.

Agent de formatage FA

Cet agent offre le service de formatage de documents avant leur restitution aux usagers. Pour ce faire, nous pouvons soit envoyer les documents sur le site distant qui offre ce service, soit utiliser un service mobile qui s'exécutera sur le site où résident les documents à formater. Le premier cas de figure peut dégrader les performances du

réseau alors que la seconde solution permet d'atteindre des performances satisfaisantes si la taille de l'agent à migrer est moins importante que la taille des documents à formater.

Agent d'expansion de requête EQA

L'agent d'expansion de requêtes peut utiliser un thesaurus pour générer une nouvelle requête qui permettra de couvrir un champ de recherche plus large que celui spécifié à l'origine. Par exemple, la requête suivante

> *L'usager désire avoir l'ensemble des images qui représentent un patient traité pour une maladie de peau. Cette maladie s'est manifestée par des tâches rouges sur le visage.*

peut être étendue à une nouvelle requête dans laquelle de nouveaux termes peuvent être introduits grâce au thesaurus, comme:

- malade / patient
- maladie de la peau / dermatologie

Les agents décrits précédemment ont été regroupés dans un sous-système pour prendre en charge les requêtes usagers. Une fois analysées et les sites pertinents identifiés, les requêtes sont dirigées vers le sous-système de stockage et d'accès aux données afin d'être effectivement exécutées.

3.2 Système de Stockage et d'Accès aux Données

Le sous-système de stockage et d'accès aux données permet d'accéder aux bases de données contenant les artefacts. Nous y intégrons l'ensemble des fonctionnalités d'indexation et de recherche des documents multimédia.

Un agent LSA est associé à un sous-système de stockage afin de gérer son intégration dans le système global. Les composantes du système de stockage sont :

- le serveur Z39.50 et son API pour permettre l'accès d'une manière standard aux bases de données mises en place dans le système,
- le moteur de recherche qui intervient après la réception de la requête par le serveur Z39.50,
- le serveur de documents qui permet de retourner les documents si l'usager désire les afficher.

Comme l'indique la Figure 3.7, le LSA crée un client Z39.50 pour pouvoir interroger les bases de données accessibles via le serveur Z39.50 et ce, pour chaque requête d'interrogation reçue. Deux types de serveurs sont utilisés, l'un pour traiter les requêtes et l'autre pour restituer les documents, ce qui permet de ne pas surcharger l'un ou l'autre des serveurs en cas d'affluence. Dans ce projet, nous nous concentrons sur la recherche des images via des descriptions textuelles ou des caractéristiques visuelles. Ainsi, le moteur de recherche se base sur deux engins de recherche, le premier utilise des attributs textuels pour retrouver une image alors que le second utilise des caractéristiques visuelles.

Dans le cadre de notre projet, nous avons adapté un serveur Z39.50 pour la recherche de documents images en intégrant notre propre moteur de recherche grâce au mécanisme d'API proposé par le CNIDR. Ce serveur est accessible grâce à un client Z39.50 développé pour la circonstance.

Les services principaux d'un serveur Z39.50 sont les suivants:

- **la connexion**: ce service est utilisé par les clients Z39.50 pour initialiser une connexion qui permettra de commencer un processus de recherche,

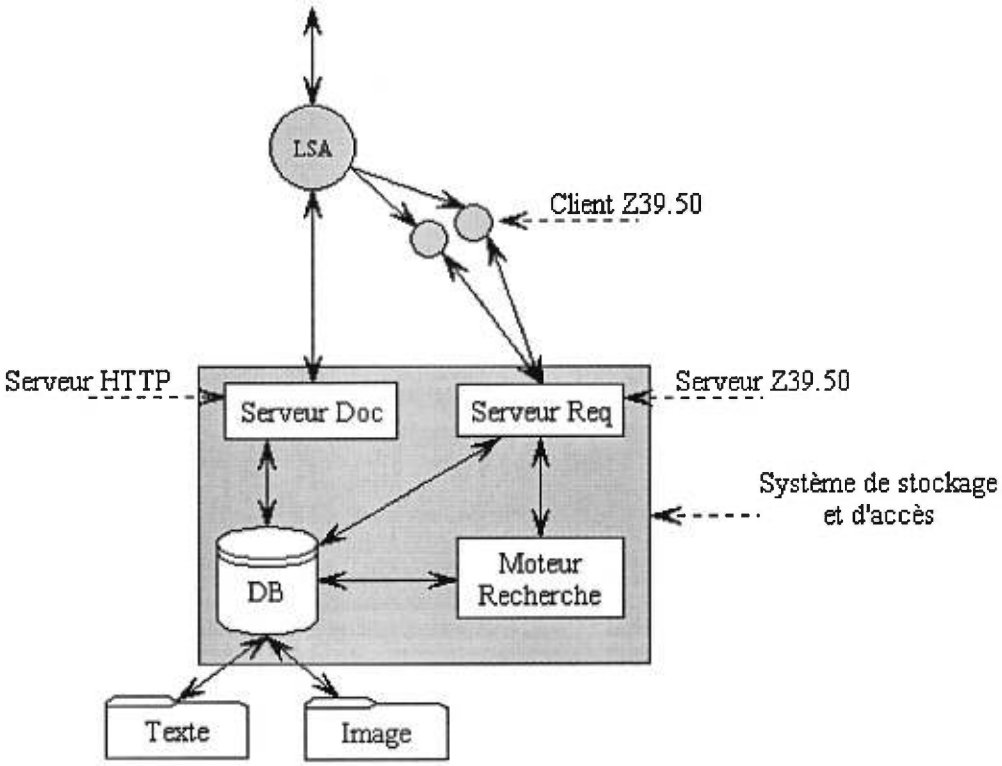


Figure 3.7 : Architecture du système d'accès et de stockage des données et son agent LSA

- **L'interrogation**: les requêtes sont soumises au serveur par le client grâce à des primitives standardisées. Ces requêtes peuvent spécifier des conditions sur tous les attributs de recherche spécifiés dans le standard Z39.50. Si les requêtes ne respectent pas la syntaxe du standard Z39.50 alors elles sont rejetées,
- **la lecture d'un document**: ce service permet aux usagers de lire le contenu des documents trouvés,
- **la navigation dans la liste de résultats**: ce dernier service est assuré par le serveur Z39.50 qui maintient la liste des résultats d'une recherche. Ainsi, l'usager peut naviguer dans la liste, supprimer de la liste les documents qui ne l'intéressent pas et éventuellement demander la sauvegarde de cette liste.

Le serveur Z39.50 ouvre une session pour chaque client conforme au standard Z39.50 qui tente de se connecter. Tous les échanges entre le serveur et les clients se font via les

services standards cités ci-dessus. Après une étape d'initialisation et de négociation, le client peut envoyer ses requêtes et attendre en retour les résultats de la recherche. C'est un échange de type synchrone qui s'établit entre les deux entités.

Le serveur est sollicité à deux niveaux dans le processus de traitement d'une requête: à l'arrivée de la requête, pour vérifier sa conformité au standard Z39.50, et à la restitution des résultats de recherche effectuée par des engins auxquels le serveur aura transmis la requête.

Cet échange d'informations est rendu possible grâce à une API qui permet le transfert des informations entre le serveur et le sous-système de recherche dans les deux sens du chemin d'exécution d'une requête.

Le moteur de recherche doit respecter un format précis pour la restitution des résultats. Il doit indiquer :

- le nombre de documents retrouvés,
- pour chaque document, la similarité, la référence et éventuellement une brève description.

L'objet client proposé possède un ensemble de méthodes lui permettant d'entamer une session d'interrogation avec le serveur, d'envoyer des requêtes et de recevoir les résultats en utilisant les services du standard Z39.50. Ce client est créé par LSA à la réception d'une requête d'interrogation.

3.3 L'interface au Système (API)

Les étapes principales de fonctionnement du système suivent le schéma suivant:

- connexion au système et identification éventuelle,
- soumission de requêtes,

- restitution de la liste des résultats,
- navigation dans la liste des résultats,
- visualisation d'un document.

Connexion au système: cette étape permet de spécifier au système les bases de données à interroger. L'identification de l'usager peut être demandée durant ce processus. Le système ne pourra traiter les requêtes usagers qu'une fois la connexion établie.

Soumission de requêtes: le système accepte des requêtes pour la recherche de documents de type image ou texte. La recherche textuelle est une recherche classique qui peut être faite selon des attributs comme un auteur, une date ou bien selon un texte libre. La recherche d'image se fait par similarité des caractéristiques visuelles. Nous pouvons remarquer que ces deux types de requêtes peuvent être combinées pour former une requête plus complexe contenant plus de précision sur la recherche à effectuer. Par exemple, on peut rechercher un document image qui contiendrait une caractéristique visuelle V dont l'auteur est A. Dans ce cas, la requête est constituée de deux parties. La première spécifie la recherche sur l'auteur A et la seconde se base sur la caractéristique visuelle V.

L'usager a deux possibilités pour spécifier les caractéristiques visuelles. En effet, il peut le faire par description textuelle, par exemple "couleur rouge". C'est l'approche la plus souvent utilisée dans la littérature. Le problème est que certaines caractéristiques visuelles sont souvent difficiles à spécifier en texte, par exemple, comment peut-on décrire la texture d'une image?

Une alternative est d'utiliser des images exemples comme une spécification approximative de la requête visuelle. L'usager doit identifier une image exemple pour trouver d'autres images similaires. C'est cette approche que nous avons adopté dans notre projet.

Soit une image modèle M dans laquelle l'usager a identifié une caractéristique visuelle V qui l'intéresse. Un exemple de combinaison de requête textuelle et visuelle est comme suit:

Requête textuelle (RT) : (Auteur = A)
Requête image (RI) : (Image exemple =M, caractéristique visuelle = V)
Requête finale: (RT, RI)

Une information indiquant le type de recherche à effectuer (textuelle ou image) est ajoutée à la requête finale. Cette information est utilisée par le système pour savoir sur quel type de document effectuer la recherche (image ou texte) et selon quel type de contenu.

Restitution de la liste des résultats: les résultats obtenus après la recherche sont restitués sous la forme d'une liste renfermant des informations sur le document trouvé, la pertinence du document ainsi que sa référence.

Navigation dans la liste des résultats: l'usager doit être capable de naviguer dans la liste maintenue par le serveur. Cette navigation peut se faire dans les deux sens, avant et arrière.

Visualisation d'un document: à n'importe quel moment, l'usager est capable de visualiser un document retourné dans la liste. Différents types de présentation sont possibles. Dans notre cas, nous choisissons HTML afin d'utiliser les navigateurs commerciaux.

3.4 Description du Modèle de Recherche

Les performances du moteur de recherche dépendent de l'organisation de ses données et du modèle de recherche utilisé pour retrouver les documents désirés par l'utilisateur. Des décisions conceptuelles ont été prises pour que l'accès et la recherche soient

rapides. Les informations de base autour desquelles s'articule le moteur de recherche sont les suivantes :

- les indexes construits à partir du contenu des images. Nous avons trois types d'index, chacun a été généré pour une caractéristique de l'image,
- l'index construit à partir des textes qui accompagnent les images,
- le planificateur de recherche.

Nous commençons par donner la structure des requêtes que l'usager pourra soumettre au système puis aborderons les stratégies de recherche utilisées pour avoir un système de recherche satisfaisant.

3.4.1 Structure d'une requête

Comme il a été mentionné précédemment, le standard Z39.50 a été utilisé pour la formulation de requêtes et l'interrogation de la base de données des artefacts image.

Comment utiliser le standard dans le cas de notre base de données d'artefacts ?

Deux aspects devront être considérés: le premier concerne la recherche textuelle et le second la recherche image.

3.4.2 La recherche textuelle

Le standard Z39.50 offre toutes les possibilités pour effectuer des recherches textuelles avancées. Dans notre cas, nous avons utilisé l'ensemble d'attributs bib1 pour générer nos requêtes. L'utilisateur pourra alors utiliser des attributs de recherche comme un nom d'auteur, un titre et un sujet, ou bien faire des recherches libres.
Comme exemple, nous avons la requête suivante:

Recherche des documents textuels dont l'auteur est Notman ou Traquair.

La requête Z39.50 générée est la suivante:

OR (Notman [1=1033], Traquair [1=1033])

où 1033 signifie auteur dans l'ensemble des attributs bib1 numéroté 1 dans le standard Z39.50.

3.4.3 La recherche visuelle

L'utilisateur utilise une recherche d'images par similarité. Cette recherche consiste à sélectionner une image parmi un groupe d'images affichées et précise la méthode d'indexation d'image selon laquelle le moteur doit faire sa recherche. Après le choix de l'image et de la méthode d'indexation, une requête contenant le nom de l'image, le type de document et le type d'index est générée. Pour ce faire, nous avons proposé une extension au standard.

Extension du standard Z39.50

Le moteur de recherche proposé ne traitant que des requêtes respectant le standard Z39.50 alors nous avons fait une extension au standard pour pouvoir prendre en compte le cas des images. L'extension a consisté à identifier un ensemble de caractéristiques visuelles que nous avons intégrées au standard Z39.50.

Nous avons proposé de regrouper les différents types de documents sous une référence DOCUMENT_TYPE (Image, Texte, ...) et le type d'indexation sous une autre référence INDEX_TYPE. Ainsi, nous avons deux ensembles dans lesquels chaque composante sera identifiée à son tour par une référence.

Si nous attribuons les références 7 et 8 respectivement à l'ensemble DOCUMENTE_TYPE et à l'ensemble INDEX_TYPE et sachant que pour ces ensembles nous avons les attributs suivants :

DOCUMENT_TYPE :

- 1001 (document texte),
- 1002 (document image),
- 1003 (document vidéo),
- 1004 (document multimédia).

où 1001, 1002, 1003 et 1004 sont les valeurs possibles de Document-type, correspondant aux différents types de documents.

INDEX_TYPE :

- 1023 pour l'index basé sur l'intensité absolue des pixels (IA_index)
- 1024 pour l'index basé sur la méthode de Marc/Hildreth (MH_index),
- 1025 pour l'index basé sur la tranformée de frourier (TF_index),

L'expression d'une requête de recherche par similarité consistera à spécifier l'image exemple, le type de document à rechercher et le type d'index utilisé dans la recherche. Dans ce cas, la requête Z39.50 générée pour rechercher les images qui ressemblent le plus à l'image référée par M1 selon le type d'index IA_index est :

M1[7=1002,8=1023]

Cette requête signifie que nous recherchons un document de type image (1002 de l'ensemble 7) dont la référence est M1 selon l'index IA_index (1023 de l'ensemble 8).

La combinaison des deux types de requêtes (texte et image) est possible grâce à l'utilisation d'opérateurs booléens.
Les deux requêtes introduites ci-dessus se combinent de la manière suivante:

AND (OR (Notman [1=1033], Traquair [1=1033]) , M1[7=1002,8=1023])

Cette requête spécifie une recherche selon des attributs textuels (noms d'auteurs) et des attributs visuels (IA_index).

3.4.4 Organisation des catalogues

Au début du chapitre, nous avons mentionné que le système se basait sur un système de catalogue à deux niveaux. Ces deux niveaux vont permettre un gain de temps dans le processus de recherche et permettront d'éviter la surcharge du réseau en diminuant le nombre de sites à solliciter inutilement.

Au niveau global, nous retrouvons les catalogues associés aux agents de recherche globale GSA. Ces catalogues sont créés à partir des informations recueillies auprès des agents LSA responsables des base de données qui existent dans le système. Parmi les informations envoyées lors du processus d'inscription d'un agent LSA nous avons:

- le type de documents (texte, image, multimédia, etc.),
- le domaine de spécialisation (science, art, etc.),
- la langue (français, anglais, etc.).

Chaque agent GSA est responsable d'un ou plusieurs agents LSA. Les informations ci-dessus vont permettre d'identifier les sites pertinents pour une requête donnée. Pour le moment, ces informations sont définies manuellement pour chaque LSA. Une détection automatique de ces informations serait incontestablement un plus (mécanisme utilisé dans certains fureteurs Web). En ce qui concerne la détection de la langue, des systèmes sont déjà en cours d'utilisation, c'est le cas de SILC développé au RALI (http://www-rali.iro.umontreal.ca/ProjetSILC.fr.html).

La détection de domaine reste cependant plus problématique. Cependant, en utilisant les mots les plus fréquents, on peut espérer dériver les principaux domaines d'une base de documents [Duda, 94]. Nous pouvons aussi penser à utiliser une base de terminologie pour le faire : Si les mots contenus dans le document sont concentrés dans certains

domaines, alors ceux-ci vont être ceux qui caractérisent le mieux cette base. Cette étude est en cours de recherche au RALI.

Au niveau local nous retrouvons les catalogues associés à chaque base de données. Ces catalogue sont gérés par les agents LSA et sont constitués de deux types d'index, un index textuel et un index image.

L'index textuel est généré à partir de toutes les informations textuelles qui accompagnent les images. Parmi ces informations, nous avons des attributs et/ou du texte libre. Pour ce faire, nous avons utilisé les librairies fournies avec le système Isite. Celles-ci permettent l'indexation de texte classique et donnent la possibilité de soumettre des requêtes booléennes.

L'index image est généré comme décrit dans la section extension du standard Z39.50. Trois types d'index ont été générés pour les images McGill [Fadi 98]. Ces trois méthodes sont décrites plus en détail dans le chapitre 4. Basé sur ces indexes, il est possible de planifier le processus de recherche de différentes manières. C'est ce que nous abordons dans la section suivante.

3.4.5 Le planificateur de recherche

Dans notre projet, nous avons dès le début pris la décision de séparer la recherche textuelle et la recherche image pour simplifier la réalisation. Ainsi, pour notre base de documents qui contient des photographies accompagnées des descriptions textuelles, nous avons établi deux moteurs de recherche, l'un pour la recherche textuelle et l'autre pour les images. La coopération de ces deux types de moteurs de recherche nécessite une bonne planification.

Le planificateur de recherche se base sur deux moteurs pour construire le plan de recherche le plus efficace. Il y a différentes stratégies de recherche et certaines sont plus efficaces que d'autres. Les plans d'action possibles sont les suivants:

Utilisation parallèle des moteurs de recherche: dans ce cas, nous obtenons deux listes de documents en résultat, la première est le résultat obtenu à partir des attributs visuels et la seconde à partir des attributs textuels. Une étape supplémentaire est nécessaire pour fusionner ces deux listes afin d'identifier les documents en commun (voir Figure 3.8). L'inconvénient de cette méthode est que les listes générées par chacun des moteurs peuvent être très grandes, ce qui peut alourdir le processus permettant de calculer l'intersection des deux ensembles de documents. Nous pouvons évaluer le temps de calcul par l'équation suivante :

$$Tc = L1 * L2 * c + Max(T_{L1}(N), T_{L2}(N))$$

où

- N est le nombre de documents dans la base
- $L1$ est l'ensemble des documents retournés par le moteur visuel
- $L2$ est l'ensemble des documents retournés par le moteur textuel
- $T_{L1}(N)$ temps nécessaire pour construire la liste $L1$
- $T_{L2}(N)$ temps nécessaire pour construire la liste $L2$
- c le temps de calcul unitaire pour la fusion

Utilisation séquentielle des moteurs de recherche: dans ce cas, le champ de recherche initial est constitué de tous les documents de la base de données; cependant, la deuxième recherche aura un ensemble réduit de documents (voir Figure 3.8) et dans ce cas, le temps de calcul est donné par l'équation suivante :

$$Tc = T_{Li}(N) + T_{Lj}(Li)) + c$$

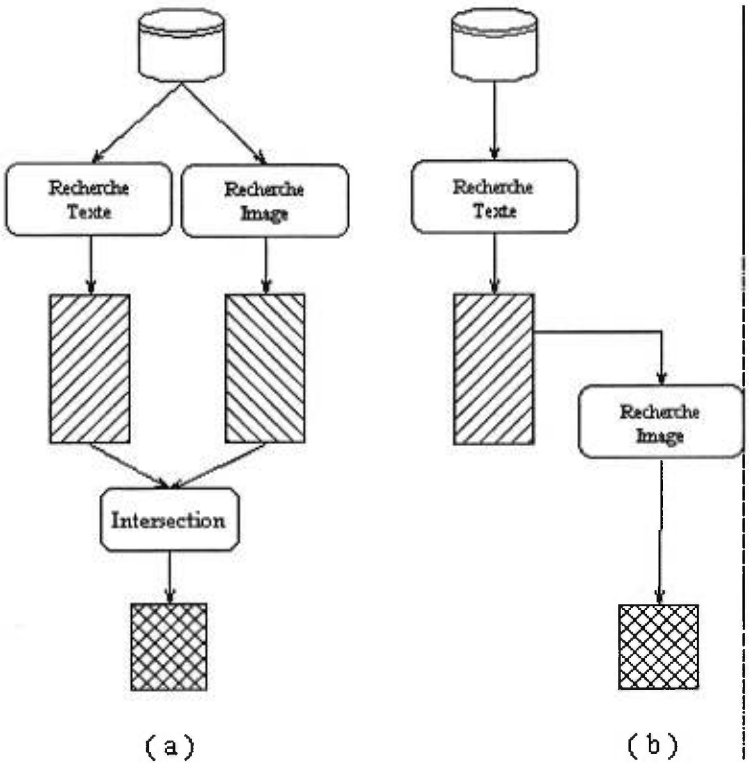


Figure 3.8 : Stratégies de recherche. (a) recherche parallèle, (b) recherche séquentielle.

Nous pouvons constater l'importance de l'ordre d'exécution des moteurs de recherche. En effet, l'évaluation de la requête textuelle dans une première étape permettra de réduire considérablement l'ensemble des documents qui seront parcourus par le moteur de recherche image.

Basé sur la deuxième stratégie, il est possible de réduire le temps de recherche des documents pertinents. Etant donné que les processus de recherche se basent sur deux étapes séquentielles, il est très important de réduire à chaque étape la taille des ensembles dans lesquels doit se faire la recherche. La stratégie adoptée se base sur les paramètres de la requête, elle permet de construire plusieurs requêtes intermédiaires. L'ordonnancement de l'exécution de ces requêtes générera des ensembles dont la taille se réduit de plus en plus proche pour enfin aboutir au résultat de la recherche demandée.

En analysant les champs de la base de données nous pouvons faire la classification suivante :

Attributs forts: ces attributs permettent une meilleure précision de recherche dans une requête; par exemple, le nom d'un auteur.

Attributs faibles: ces attributs permettent de compléter les informations principales utilisées dans la recherche comme par exemple une date de parution et la langue dans laquelle est écrit le document.

Basé sur ces deux types d'attributs, il est possible de formuler d'une autre façon la requête soumise par l'usager en considérant dans un premier temps les attributs forts puis ceux de moindre poids. De cette manière, nous réduisons lors d'une première passe l'ensemble de recherche des documents.

3.5 Scénario d'Utilisation du Système

Nous décrivons dans cette section un exemple complet d'utilisation du système, depuis la génération de la requête à partir de l'interface usager jusqu'à la restitution des résultats.

Le traitement d'une requête passe par plusieurs étapes et implique différents types d'agents dans le processus de recherche. La Figure 3.9 donne une illustration du cheminement de la requête à travers le système. Chaque étape y est indiquée.

Étape 1:

L'usager spécifie la requête de recherche à l'aide de l'interface du système. Un exemple de requête serait :

Rechercher toutes les photos des photographes Notman et Traquair dont la luminosité des pixels se rapproche le plus de l'image M1

Dans le cas de notre requête nous recherchons des documents de type image dont les caractéristiques visuelles basées sur IA_*index* se rapprochent le plus de l'image M1. L'usager utilise les différents panels de l'interface graphique du système pour générer la requête au standard Z39.50. Une fois la requête spécifiée et traduite dans le standard Z39.50, un objet requête (OBR) est créé. Il contient la requête et ses options.

Parmi les options nous avons :

- la langue à utiliser,
- le type de documents recherchés (image, texte,...),
- le domaine concerné (art, science, ...),
- ...

L'objet requête créé est envoyé à l'agent QA, ceci nous amène à l'étape 2.

Étape 2:

L'agent QA est créé une fois que l'usager soumet une requête. L'objet OBR reçu par le QA est prétraité en fonction des options utilisées par l'usager lors de la spécification de la requête. Parmi les traitements nous avons la traduction de la requête dans une autre langue grâce à l'agent de traduction TA.

L'agent QA récupère la liste d'agents GSA actifs à contacter pour la prise en charge de la requête, cette liste étant disponible au niveau de l'agent superviseur. Les résultats seront restitués à l'usager via un objet cache. Celui-ci est créé par l'agent QA et sa référence est transmise avec l'objet requête aux agents GSAs. Une fois ces opérations terminées, l'agent QA termine son exécution.

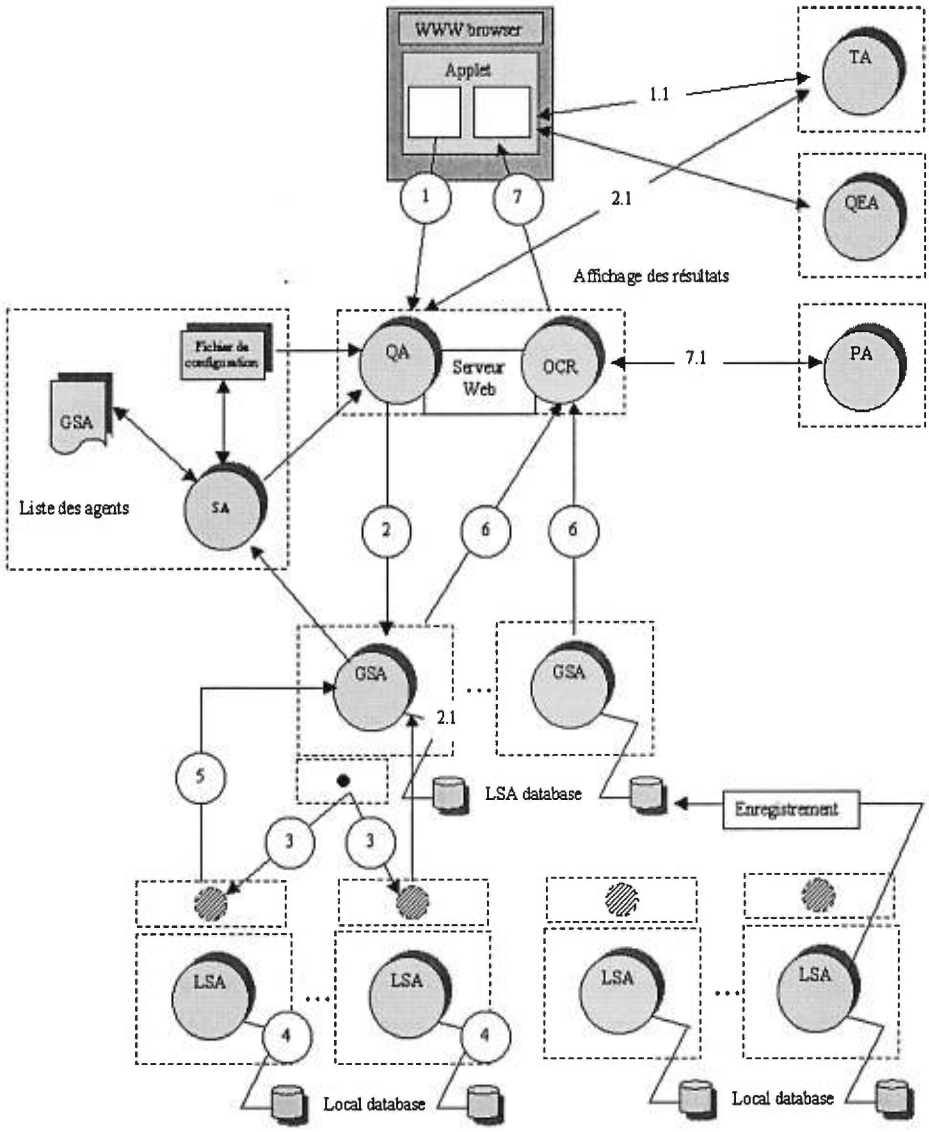


Figure 3.9 : Les différents étapes de traitement d'une requête.

Étape 3:

Lorsqu'un agent GSA reçoit un objet OBR, il l'analyse et extrait toutes les informations lui permettant de construire une liste de sites pertinents à partir du catalogue global. Chaque site est représenté par un agent LSA. Le GSA crée pour chaque LSA à contacter un processus temporaire dont la tâche est de prendre en charge la suite de l'exécution de la requête. Les résultats récupérés par ces processus sont remis à un processus commun pour un traitement ultérieur.

Étape 4:

À cette étape du processus de recherche, les agents LSAs sollicités par un GSA se connectent aux serveurs Z39.50 desservant les bases de données trouvées à l'étape 3. Les agents LSA renvoient les résultats à l'agent GSA qui les a sollicité par l'intermédiaire du processus commun créé à l'étape 3.

Étape 5:

Les résultats récupérés par les processus sont acheminés aux GSA. Les processus terminent leur exécution une fois cette opération terminée.

Étape 6:

Le GSA fusionne les listes de résultats en se basant sur le facteur de similarité accompagnant le descriptif de chaque document trouvé. Une fois la liste finale formée, elle est envoyée à l'objet cache créé à l'étape 2 qui se chargera de la restituer à l'usager. Le clone associé à l'agent GSA termine alors son exécution.

Étape 7:

L'objet cache gère la liste des résultats de la recherche. Il permet à l'usager de choisir un document en particulier dans la liste et de l'afficher. Si l'usager choisit une présentation de document particulière alors l'objet cache fait appel à l'agent PA qui offre le service demandé. Le document formaté est alors restitué à l'interface usager pour être affiché.

3.6 Amélioration de la Stratégie de Recherche

Plusieurs moyens existent pour améliorer le moteur de recherche utilisé pour l'implantation du prototype. Nous les résumons ci-dessous :

- Classification des documents selon les caractéristiques visuelles
- Utilisation d'un thesaurus pour l'expansion de requêtes
- Utilisation d'un interpréteur pour l'expansion de requête

La classification

La classification des images est l'un des moyens qui permettrait d'améliorer le temps de recherche des documents. Plusieurs techniques existent, parmi celles-ci nous pouvons citer l'algorithme LGB [Gray 82]. Cette technique permet de réduire la taille de l'ensemble de recherche des documents visuels en le décomposant en sous-ensembles dans lesquels sont regroupées des images qui respectent plus ou moins un seuil de similarité. Ainsi, le temps de recherche s'en trouve amélioré. En effet, le processus de recherche ne se base plus directement sur la distance entre les images mais sur la probabilité qu'une image appartienne à un sous-ensemble, donc plus cette probabilité est grande plus les images qui se trouvent dans cet ensemble sont de bons candidats à une solution en évitant l'inconvénient de parcourir l'ensemble des images de la base.

L'interpréteur de requête

La troisième manière d'améliorer le processus de recherche est d'utiliser un interpréteur de requêtes. Par interprétation nous désignons l'association d'une certaine sémantique à l'image. Une requête textuelle ou visuelle peut renfermer un certain nombre d'informations, qui une fois interprétées, permettront d'étendre une requête. A titre d'exemple, nous pouvons citer le cas où l'usager recherche une image qui contient un paysage, cela sous-entend que la couleur recherchée dans l'image est la couleur verte et cette information une fois interprétée permettra de générer une requête visuelle qui recherche les images qui contiennent un pourcentage de vert.

Le processus inverse est possible. Par exemple, si l'usager recherche toutes les images qui contiennent la couleur bleue, alors une requête textuelle peut être générée dans laquelle on utilisera les termes qui sont en relation avec cette couleur comme la mer, le ciel, etc. Bien que cette option soit très attrayante, elle reste très difficile à réaliser. En effet, il n'existe pas encore de système qui implante cette interprétation de façon satisfaisante.

3.7 Conclusion

Dans ce chapitre a été abordée la conception du prototype du système de bibliothèque numérique. Trois parties ont été étudiées : l'interface graphique, le système multi-agents et le système de stockage et d'accès.

Nous avons axé notre description sur les deux dernières parties car elles constituent la base du prototype. Dans la première partie a été décrit le système à agents de service. L'aspect nouveau est la hiérarchisation des agents de recherche. Dans la seconde partie, nous avons montré comment a été organisée la base de données Traquair et en quoi a consisté l'extension du standard Z39.50 au problème de recherche d'images.

Plusieurs moyens techniques s'offrent à nous pour la mise en place de la solution technique. Dans le chapitre suivant, nous faisons le point sur les choix effectués et comment ils ont été implantés.

Chapitre 4

4 Architecture Multi-Agents à Extension de Services

- Implantation -

Nous décrivons dans ce chapitre les techniques et outils utilisés pour implanter le prototype du système multi-agents dont la conception a été détaillée dans le chapitre précédent. Nous nous intéresserons aux trois sous-systèmes identifiés pour mettre en œuvre le système de bibliothèque numérique en l'occurrence l'interface graphique, le système multi-agents et le système de stockage et d'accès aux données.

Les principales décisions prises lors de l'implantation du prototype concernent:

- Les plates-formes cibles,
- L'environnement d'exécution,
- Les langages utilisés,
- Les outils et techniques utilisés pour implanter l'aspect distribué,
- Le standard utilisé pour assurer l'interopérabilité et l'uniformisation du processus d'interrogation et d'accès aux différentes collections.

Nous décrivons chacun de ces points en faisant référence à chaque partie concernée du système.

L'environnement pour lequel est destiné le prototype est un environnement distribué, en l'occurrence Internet. Le système s'utilise à travers un fureteur web (Internet Explorer ou Netscape) grâce à une Applet Java et les différents éléments qui le composent seront disséminés à travers plusieurs machines du réseau.

Mentionnons certaines limites dans cette implantation:

- Utilisation d'une seule base de données d'images accompagnées d'annotations textuelles,
- Utilisation de moteurs de recherche séparés pour le texte et l'image,
- Utilisation d'un seul type de formatage de document: HTML

4.1 Les Langages Utilisés

Différents langages de programmation ont été utilisés. Le choix dépendait non seulement des parties du système à développer, mais aussi des systèmes sous-jacent utilisés. Dans ce qui suit, nous donnerons pour chaque partie réalisée le ou les langages impliqués.

Le Système d'Interface Graphique

La nécessité d'utiliser le système à travers le Web nous a limité le choix des langages et technologies à utiliser pour la réalisation de l'interface. Nous avons opté pour la construction d'une Applet (en Java) en utilisant l'API SWING de SUN. Le seul inconvénient qui peut se poser dans l'utilisation d'Applets est la lenteur du chargement de l'application à travers le Web due aux classes qui composent l'application à exécuter. Cette solution présente néanmoins les avantages suivants :

- Interface graphique plus conviviale et plus riche,
- Système moins alourdi car l'interface de l'application s'exécute sur la machine cliente,
- Intégration facile d'objets Java distribués.

Le Système Multi-agents

Les agents du système de traitement de requêtes ont été développés en Java au dessus de l'environnement de programmation distribué HORB (Hirano Object Request Broker) [Hirano 96]. Certains agents particuliers ont été développés en C dans un souci de performance. En effet, Java présente l'inconvénient d'être un langage lent. Cependant, les avantages offerts restent toujours plus attractifs par rapport à cet inconvénient.

Le Système de Stockage et d'Accès

L'indexation et la recherche de documents textuels ou images sont prises en charge par le système Isite [CNIDR]. Ce choix a été essentiellement motivé par l'aspect coût, performance et support offert aux usagers du système. En effet, Isite est un système gratuit, dans le cas des applications non commerciales, qui implante les fonctionnalités les plus importantes énoncées dans les spécification du standard Z39.50; de plus, il fournit un ensemble de fonctions d'indexation et de recherche de texte.

Le moteur de recherche qui se base sur les engins de recherche (texte et image) a été développé en langage Perl. Le choix du langage a été fait en particulier pour faciliter le traitement des fichiers textuels obtenus en sortie des deux engins de recherche.

Pour ce qui est de l'accès, nous avons utilisé une implémentation du protocole Z39.50. Une version du serveur était codée en C alors que le client a été développé en Java (voir Figure 4.1 page suivante).

4.2 Systèmes et Standards Utilisés

Le premier point important de cette section concerne le système distribué qui a permis la réalisation de tout l'aspect réparti du prototype: HORB. Le second point est en rapport avec les standards sur lesquels nous nous somme basés: le standard Z39.50 et le protocole TCP / IP.

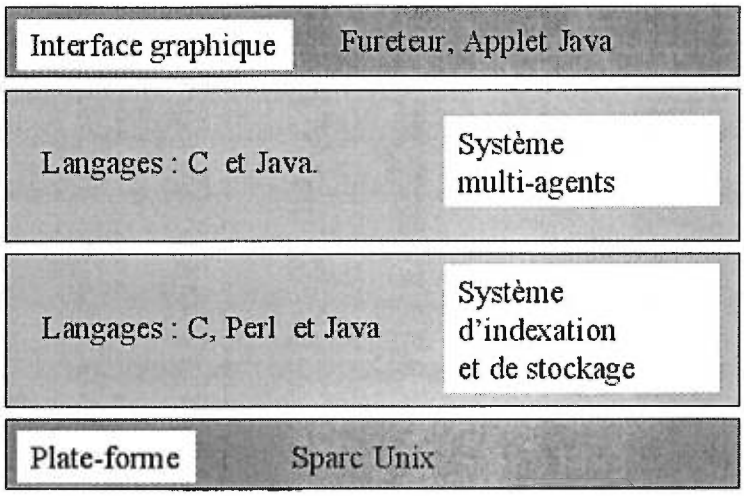


Figure 4.1 : Les trois systèmes et les langages utilisés pour l'implémentation.

4.2.1 Le Système à Objets Distribués HORB

HORB est un système en langage Java qui offre une API aux usagers pour la programmation d'applications distribuées portables. Grâce à ce système, nous faisons abstraction de l'aspect distribué d'une application en utilisant des objets distants. Il supporte le développement d'applets, les architectures clients/serveurs et présente certaines caractéristiques de mise en œuvre comme la persistance d'objets, le stockage d'objets distants, la sécurité et la programmation Web.

Il a été conçu en langage Java conforme au JDK1.1.3 de SUN (Figure 4.2 - a) et est fourni avec un compilateur HORBC utilisé pour générer les objets squelette et proxy.

Deux types d'objets sont utilisés pour la conception d'une application distribuée du type client / serveur: l'objet "proxy" et l'objet "squelette". Comme l'indique la Figure 4.2-b, l'objet proxy est créé sur le site où s'exécute le client alors que l'objet squelette est créé sur le site où se trouve le serveur.

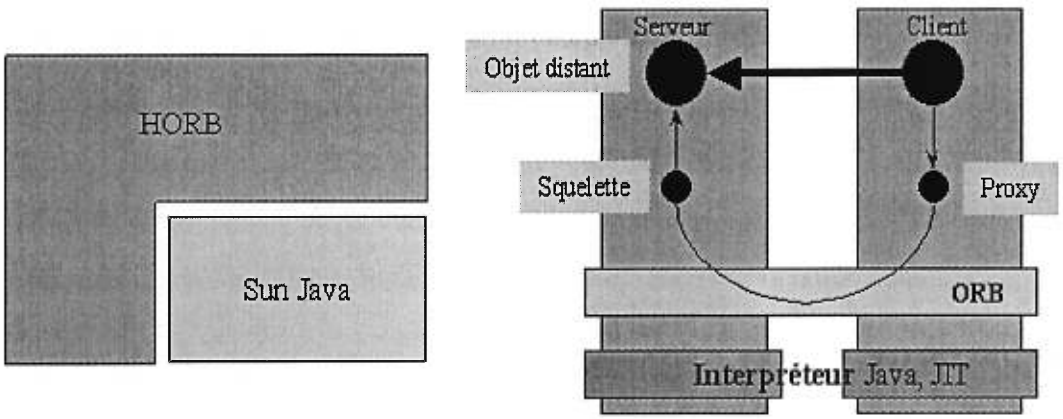


Figure 4.2 : (a) architecture de HORB, (b) fonctionnement de HORB

- l'objet proxy simule le fonctionnement de l'objet serveur distant,
- l'objet squelette est la contre partie de l'objet proxy,
- ORB (Object Request Broker) est le mécanisme de communication entre les différents objets.

A chaque proxy est associé un objet squelette permettant une utilisation locale des objets qui se trouvent sur un site distant. En effet, tout appel à une méthode au niveau de l'objet proxy est dirigé vers l'objet squelette pour être exécuté. Le résultat est retourné à l'objet proxy qui lui à son tour le restitue à l'appelant.

La mise en place de l'application client/serveur consistera à compiler les classes Java client et serveur à l'aide du compilateur HORBC afin de générer des objets distribués qui vont être utilisés en faisant abstraction totale des moyens de communication réseaux nécessaires.

4.2.2 Les Standards Utilisés

Le standard Z39.50

Nous avons utilisé le standard Z39.50 pour l'interrogation des bases de données et pour permettre l'uniformisation de la syntaxe des requêtes. Différentes implémentations du standard ont été proposées; dans notre cas, nous avons étendu celle du CNIDR à la recherche d'images.

Le protocole TCP / IP

Certains agents ont été développés en langage C, c'est le cas de l'agent de traduction de requête. L'ensemble des communications avec cet agent ont été implémentées au dessus du protocole TCP / IP grâce au mécanisme de socket. La Figure 4.3 nous montre à quel niveau interviennent le système et les standards utilisés. Le système multi-agents fait appel à HORB ou directement au protocole TCP/IP. L'accès aux bases de données n'est possible que via le standard Z39.50.

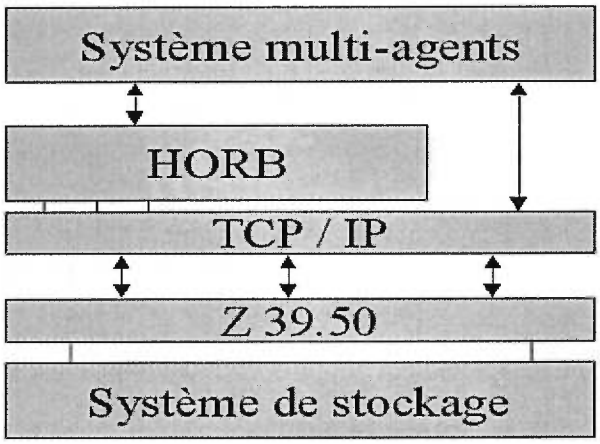


Figure 4.3 : Système et standard utilisés

4.3 Implantation du Système Multi-agents

Le système a été conçu pour être installé sur plusieurs machines interconnectées par le réseau Internet. Un serveur HORB est mis en fonction à l'installation du système sur

chaque machine qui doit desservir des agents, ces serveurs vont desservir les agents de base SA (Supervisor Agent), QA (Query Agent), OC (Object Cache), GSA (Global Search Agent) et LSA (Local Search Agent).

4.3.1 Création des agents de base

La création des agents se fait du plus haut niveau dans la hiérarchie pyramidale vers le plus bas niveau, c'est à dire SA puis GSA et finalement LSA. Ceci parce que la création des agents GSA et LSA utilise l'agent SA et celle de LSA se base sur l'agent GSA.
Imaginons la situation distribuée suivante:

- L'agent SA est créé sur la machine M_SA,
- Un agent GSA(i) est créé sur la machine M_GSAi ,
- Un agent LSA(i) est créé sur la machine M_LSAi,

Les étapes de la mise en place des agents du système sont indiquées ci-après.

Étape 1 : créer l'agent superviseur sur la machine M_SA.

Le système HORB permet de déclarer plusieurs agents sur une seule machine. Dans notre cas, le serveur HORB est lancé avec en paramètre un fichier de configuration SA_services.cfg grâce auquel nous avons déclaré l'agent SA et les services qu'il offre. Dans ce cas, nous avons les informations suivantes:

- L'agent superviseur SA
- Les services AddToGSAL, GetGSAL, RemoveFromGSAL

Une fois le serveur HORB lancé, il se met à l'écoute de clients qui eux doivent indiquer dans leurs requêtes le nom de l'agent avec lequel ils veulent communiquer et le service à utiliser.

Étape 2 : créer les agents GSA sur les différentes machines M_GSAi

De même que pour l'agent SA, un serveur HORB est aussi lancé sur la machine ou doit être créé le GSA. Ce serveur se met à l'écoute des clients désirant faire appel aux services d'un agent GSA.

Étape 3 : installer les agents LSA sur les différentes machines M_LSAi

Le serveur HORB installé sur la machine M_LSAi va permettre de créer un agent LSA à chaque requête soumise par les GSA. Le LSA est installé avec la base de données qu'il aura à gérer.

Étape 4 : installer le serveur HORB des agents QA et OC sur la machine M_QA_OC

le serveur HORB permettant de créer les agents QA et OC devra être installé sur la même machine d'où provient l'Applet et ceci est dû à un mécanisme de sécurité instauré au niveau des fureteurs Web.

4.3.2 Utilisation du système HORB

HORB est utilisé grâce à l'API Java incluse dans le système. Toutes les classes à utiliser dans l'application distribuée sont compilées à l'aide du compilateur Horbc afin de générer le proxy et le squelette correspondant. A titre d'exemple, nous aurons pour l'agent SupervisorAgent:

- *La classe SupervisorAgent,*
- *Le proxy SupervisorAgent_Proxy.class,*
- *Le squelette SupervisorAgent_Skeleton.class,*

La classe SupervisorAgent implante tous les services de l'agent SA, parmi ces services, nous avons l'enregistrement, la consultation et le retrait d'un agent.

Les serveurs HORB seront utilisés pour permettre l'accès aux instances des objets distants. Nous montrons par deux exemples typiques le mécanisme mis en œuvre. Le premier exemple est celui de l'enregistrement d'un agent GSA auprès de l'agent SA et le second est la création d'un agent QA.

Exemple 1 : enregistrement de l'agent GSA

Un objet démon est créé à l'installation de l'agent SA. Le serveur HORB permet l'accès aux services du superviseur SA à chaque client qui en fait la demande par le moyen d'une connexion. Dans ce cas, le serveur HORB crée un Thread SA associé au démon et celui-ci prendra en charge l'ensemble des communications avec le client. Dans notre cas, GSA est le client qui va créer une instance de l'objet SA pour pouvoir s'enregistrer (voir Figure 4.4).

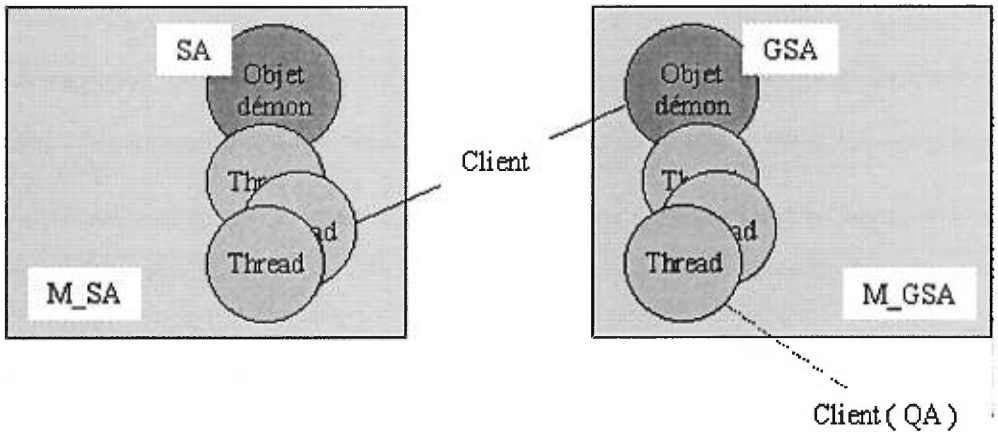


Figure 4.4 : Exemple d'objet thread SA créé à la connexion d'un agent GSA pour l'enregistrement

La Figure 4.4 montre que l'objet démon GSA se connecte à l'agent SA, un thread est alors créé. Ce thread terminera son exécution après l'enregistrement de l'agent GSA. Les threads de GSA créés à la connexion des agents, par exemple QA, n'enregistreront pas une nouvelle fois GSA.

La méthode *doRegistrationToSA (AserverObjet server)* de la classe *GlobalSearchAgent.java* fait l'enregistrement de l'agent GSA auprès de SA. C'est une méthode qui n'est appelée qu'une seule fois à la création de l'agent GSA. Le code de la méthode est le suivant:

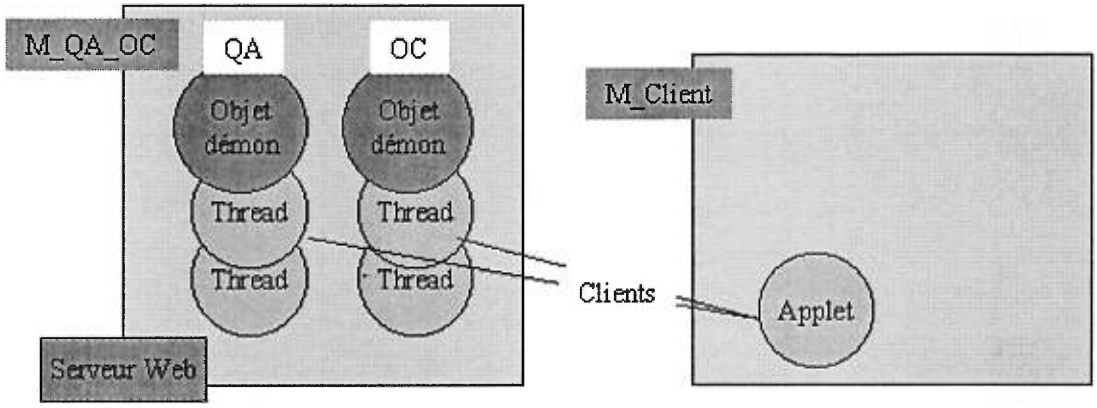
```
1 Private boolean doRegistrationToSA( AServerObject server ){
2 try{
3       SupervisorAgent_Proxy sa =
4       new SupervisorAgent_Proxy(new HorbURL("horb://"+server.name+":"+server.port +
5       "/" + "AddToGSAL"));                    // Crée SA sur le site distant
6       sa.addToGSAL( gsaObjectServer.name, gsaObjectServer.port ); // Enregistre GSA
7       sa._release();                          // ferme la connexion avec SA.
8  }catch(HORBException e){RepportException(e);}
9 } // fin méthode
```

Description :

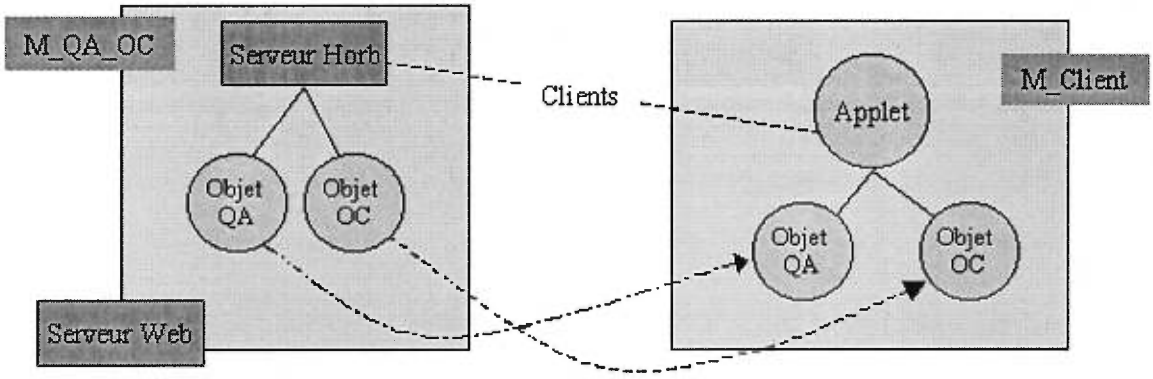
Ligne 3 :	Création de l'objet proxy de l'agent SA. Les paramètres sont l'adresse IP de la machine M_SA et le port d'écoute du serveur horb ainsi que le nom du service à utiliser.
Ligne 6 :	Enregistrement de l'agent GSA.
Ligne 7 :	Fermeture de la connexion. Le thread lancé sur la machine M_SA termine son exécution.

Exemple 2 : Création des agents QA et OC

Dans cet exemple, nous montrons comment se fait la création dynamique des agents QA et OC. Comme nous l'avons déjà mentionné, il y a un serveur Horb lancé sur la machine M_QA_OC où se trouve le serveur de pages Web et sur laquelle sont créés des démons pour chaque agent QA et OC. La création des objets proxy de QA et OC est faite au niveau de l'applet, cette création fait appel à l'HORB comme nous l'avons vu pour l'agent GSA.



- a -



- b -

Figure 4.5 : (a) Création des objets QA et OC et exécution sur le site serveur. (b) les objets QA et OC sont envoyés sur le site client.

Comme amélioration, nous pouvons proposer de créer ces deux objets sur la machine M_QA_OC puis les envoyer vers la machine où s'exécute l'applet. Cette technique permettra d'éviter de surcharger la machine sur laquelle s'exécute le serveur Web et ainsi prendre un plus grand nombre de requêtes usagers. La Figure 4.5 résume les étapes de création des agents QA et OC (Figure 4.5 a) et montre la variante qui se base sur la mobilité des objets QA et OC vers le site client (Figure 4.5 b). Les objets mobiles QA et OC sont créés par le serveur HORB sur la machine M_QA_OC puis envoyés sur la machine M_Client pour être exécutés; ainsi, ce sont les ressources de la machine cliente qui sont utilisées pour la prise en charge d'une requête. Seule la première solution a été implantée dans le cadre de notre projet. La même démarche est utilisé pour créer les objets distribués implantant les agents du système.

4.4 Implantation du système d'accès et de stockage

L'implantation du système de stockage a consisté à :

- configurer le serveur Z39.50,
- développer le client Z39.50,
- créer la base de données,
- développer le moteur de recherche.

La mise en place du système consistera alors à lancer le serveur Z39.50 afin de prendre en charge les connexions de clients Z39.50.

4.4.1 Le serveur Z39.50

Le serveur Z39.50 utilisé est celui proposé dans le système Isite conçu par le CNIDR. Ce système dispose en outre d'outils qui permettent de faire de l'indexation et de la recherche textuelle et d'une API permettant sa configuration. Dans notre cas, nous l'avons adapté pour pouvoir utiliser notre propre moteur de recherche à la place de celui proposé par défaut.

La spécialisation du système Isite à été possible grâce à la définition des fichiers de configuration zserver.ini et sapi.ini. Le premier fichier contient les paramètres du serveur, par exemple le port d'écoute, le nombre de connexions maximales et le deuxième fichier est utilisé pour permettre au serveur de savoir quel engin de recherche utiliser pour la recherche et quelle application utiliser pour lire le contenu d'un document. Dans la section suivante nous décrivons le contenu de chacun des fichiers.

Description des fichiers de configuration

1- Le fichier zserver.ini

 # Le niveau de détail des message affichés (de 0 à 9)

DebugLevel=5

 # Permettre la création d'un processus à chaque connexion de client. ServerType=INETD

 # Le numéro de port utilisé

Port=6668

 # Le nombre maximal de sessions

MaxSessions=50

 # Le nom du fichier de configuration API

SAPI=/home/vautour4/hamard/My-Database/sapi.ini

Le fichier log des accès

AccessLog=zserver_access.log

La liste des bases de données accessibles. Dans notre cas, nous n'avons qu'une seule base de

données, Traquair.

DBList=Traquair

2- Le fichier sapi.ini

DBList=Traquair

[Traquair]

Type=SCRIPT

Location=/home/vautour4/hamard/Projet/Moteur/Main-Engine.pl

GetFull=/home/vautour4/hamard/Projet/Moteur/GetFull.pl

Results=/tmp/Z39/results

Dans ce fichier ont été mentionnés les chemins d'accès au moteur de recherche mixte textes et images, (Main-Engine.pl) et à l'application qui permet de lire le contenu d'un document (GetFull.pl). Ainsi, lorsque le serveur Z39.50 reçoit une requête, il la transmet au moteur de recherche et après son traitement, les documents trouvés sont retournés grâce à l'application réservée à cet effet.

4.4.2 Le client Z39.50

Le client Z39.50 (Zclient.java) a été développé grâce à une API java qui implémente l'ensemble des services introduits lors de la description du standard Z39.50. La classe Zclient.java possède les principales méthodes suivantes :

Connexion au serveur

Méthode : *Public void doConnect(String host, int port)*

Description: Méthode qui permet de se connecter au serveur Z39.50 après une phase de négociation de paramètres entamée par le client. Si le serveur accepte alors une session est initiée.

Les services du standard Z39.50 utilisés dans cette méthode sont :

init.Request(0,"yyyyyyy",5000,10000,client.login,client.password,"",null,true);

init.Response(connection.doRequest(req, client.host, client.port));

Les paramètres des deux méthodes ci-dessus sont décrits dans l'annexe B réservée à la description du standard Z39.50.

Envoi de requêtes de recherche

Méthode : *Public boolean doFind(String query)*

Description : Méthode qui permet d'envoyer une requête Z39.50 au serveur Z39.50.

Les services du standard Z39.50 utilisés sont :

search.Request(referenceId, smallSetUpperBound,
largeSetLowerBound,
mediumSetPresentNumber,
replaceIndicator,
(MustUseDefault?resultSetName:"1"),
databaseName,
smallSetElementSetName,
mediumSetElementSetNames,
preferredRecordSyntax,
query,
query_type,
searchResultsOID,
Z39attributesPlusTerm);

search.Response(connection.doRequest(req, client.host,client.port));

Lecture des résultats

Méthode : *Public boolean doGetResult(int resultSetStartPoint, int numberOfRecordRequested*

Description : Méthode utilisée pour la lecture des enregistrements trouvés. On indique le rang du premier enregistrement et celui du dernier enregistrement.

Les services du standard Z39.50 utilisés :

present.Request(0, (MustUseDefault?resultSetName:"1"),
resultSetStartPoint,
numberOfRecordRequested,"F",
preferredRecordSyntax);

Affichage des résultats

Méthode : *Public void doDisplayRecord(Z39present present, int refRecord)*

Description : Méthode qui permet de récupérer un enregistrement au complet.

Lecture de la forme résumée des documents trouvés

Méthode : *Public Vector doDisplayHits(int start, int count)*

Description : Méthode utilisée pour récupérer seulement la description sommaire des documents retrouvés.

4.4.3 Description de la base de données

Dans cette section, nous verrons comment a été mise en place la base de données et étudierons la structure d'un document, les outils d'indexation des documents textuels et images ainsi que l'extension du standard Z39.50 à la recherche d'images.

La base d'images qui est utilisée pour concevoir le prototype de démonstration du projet de bibliothèque numérique dans lequel s'insère notre projet de maîtrise est la collection "Ramsay Traquair". Elle fait partie de la collection canadienne d'architecture et elle est constituée de photographies numérisées d'édifices architecturaux du Québec et d'objets en argent ayant servi à des fins domestiques ou religieuses. Ces images, qui représentent une grande partie de l'histoire de l'architecture au Québec, proviennent principalement de l'œuvre de Ramsay Traquair : architecte, enseignant et directeur de l'école d'architecture de l'université McGill de 1913 à 1939. Ces images représentent également les travaux de collaborateurs de R. Traquair (E. Gariépy, G. A. Neilson, M.E. Massicotte) et de studios de photographies de l'époque (Livernois Ltd , Notman Compagny).

Description de la base de données

La base Traquair, actuellement disponible sur le WWW à l'adresse http://cac.mcgill.ca/traquair, est constituée de 4700 photographies. Les recherches proposées aux utilisateurs de ce site sont des recherches par états conditionnels (par mots-clés) réalisées à partir de certains attributs comme le titre, l'adresse du monument ou la date à laquelle a été prise la photographie.

Dans le cadre de notre projet, la base de données ne contient qu'une centaine de photos pour lesquelles nous avons complété la description textuelle initiale par un ensemble de caratéristiques, à savoir, la dimension de l'image, la taille, le type d'image et les différents types d'indexation visuelle utilisés. Les informations textuelles sont utilisées pour la génération de l'index textuel. Les caractéristiques visuelles de l'image sont extraites pour former les index visuels. Nous en utilisons trois dans le cadre de notre étude.

Un exemple est donné sur la page suivante (Figure 4.6). Sa description est comme suit:

<TITLE> Windmill/Lower Lachine Road, Montréal</TITLE>
<COLLECT> Architecture </COLLECT>

<IREF> 100857 </IREF>

<STREET> </STREET>

<CITY> Lachine </CITY>

<PROVINCE> Québec </PROVINCE>

<MJTYPE> Industrial </MJTYPE>

<MNTYPE> Farm </MNTYPE>

<PHTYPE> Building </PHTYPE>

<PHOTOGRAPHER>Notman</PHOTOGRAPHER>

<PICDATE></PICDATE>

<BUILDTDATE></BUILDTDATE>

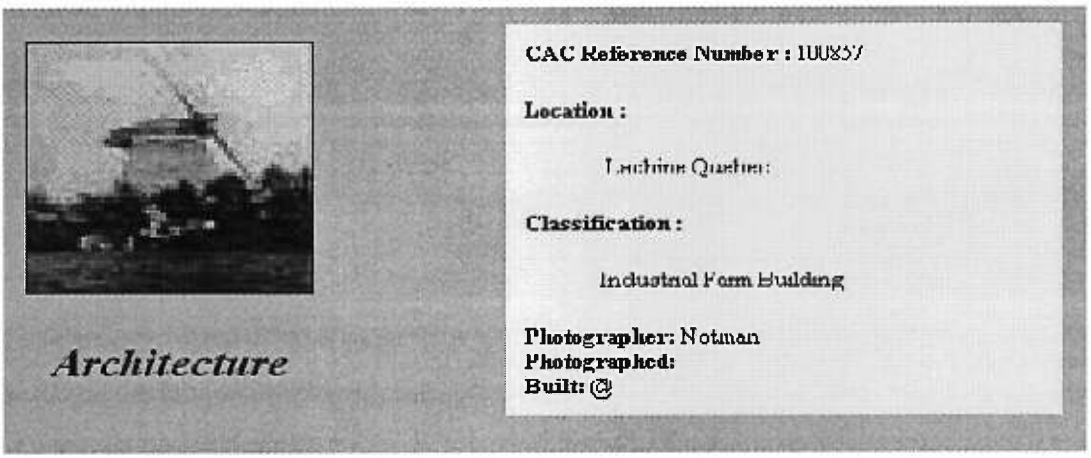


Figure 4.6 : Un exemple de document image contenu dans la base

Les attributs ci-dessus sont des attributs textuels IREF réfère à l'image correspondante.

Ceux utilisés pour la description de la partie visuelle sont:

<DIM>60x60</DIM>

<SIZE>1.5 </SIZE>

<TYPE>Jpg </TYPE>

<INDEX> l </INDEX>

<INDEX>z</INDEX>

<INDEX>m</INDEX>

Description de l'indexation des documents

À partir des informations fournies dans la base de données existante, nous avons défini une nouvelle structure de document en ajoutant des champs prenant en compte des caractéristiques visuelles. Des TAGS ont été utilisés pour chaque propriété du document comme c'est décrit dans le tableau ci-dessous:

Tableau 4.1: Tag associés à un document image.

Propriété	**TAG**	**Description**
Numéro de référence	*<IREF>*	Le numéro de référence de l'image numérique dans la base.
Titre	*<TITLE>*	Le titre de la photographie.
Adresse	*<STREET>* *<CITY>* *<PROVINCE*	Les informations sur la localisation (rue, numéro de rue, ville, province) de l'objet de la photographie.
Date de l'image	*<PICDATE>*	La date de création de l'image numérique.
Major_Type	*<MJTYPE>*	Le type principal de l'objet représenté dans l'image comme, Par exemple, les types d'édifices commerciaux et religieux.
Minor_type	*<MNTYPE>*	Le type secondaire de l'objet représenté dans l'image. par exemple, banque, restaurant, église, mosquée, etc.
Photographe	*<PHOTOGRAPHER>*	Le nom du photographe.
PHDATE	<PHDATE>	La date de la photographie.
PHTYPE	<PHTYPE>	La catégorie laquelle a été assignée l'image dans la base telles que les classes : édifice, objet, vue, événement.
Collection	<Collection>	La collection (architectural ou objet en argent) dont fait partie l'image . Par exemple, la collection d'architecture ou la collection d'objet en argent.
Dimension	<DIM>	La taille en pixels du document numérique
Taille	*<SIZE>*	La taille (en pixels) de l'image numérique.
Type	<TYPE>	Le format du fichier numérique.
Type d'index visuel	*<CARACV>*	Type d'index visuel: M, Z ou L

1 - Indexation du texte

Les métadonnées citées dans la section précédente seront utilisées par l'outil Iindex de Isite pour générer les indexes de la base de données. Ainsi, l'usager pourra spécifier dans sa requête les champs selon lesquels il veut faire sa recherche. Tous les champs sont utilisés par le moteur d'indexation y compris le champ de la caractéristique visuelle recherchée CARACTV. Les valeurs de ces champs sont textuelles, ce qui nous permet d'utiliser Iindex pour leur indexation et recherche. Par exemple un nom d'auteur, un titre, etc. L'indexation selon CARACTV permettra un gain de temps dans le cas où la caractéristique visuelle demandée ne figure pas dans la liste des caractéristiques extraites de l'image. En effet, si l'usager spécifie dans sa requête qu'il recherche une image dont

la caractéristique c1 est x et que cette caractéristique n'a pas été extraite de l'image, alors la recherche textuelle permettra de déduire rapidement qu'il n'y a pas de réponse, sans faire de recherche sur l'attribut visuel.

2 - Indexation de l'image

Les images, regroupées dans un répertoire différent de celui des documents textuels, sont indexées de trois façons [Fadi 98]. Chaque type d'index est créé à partir d'une caractéristique visuelle précise de l'image. Le premier type d'index se base sur la valeur de l'intensité absolue des pixels d'une image, cet index est noté IA_*index (IA :Intensité Absolue)*. La recherche basée sur cet index n'est pas très performante.

Les images pouvant être très complexes, il devient alors difficile d'utiliser une méthode aussi simple que IA_index et espérer avoir de bons résultats dans la recherche des images. Les travaux menés à l'université McGill [Fadi 98] ont montré l'utilisation de différentes techniques d'indexation en fonction du type d'images, complexes ou simples. Dans les deux cas, les techniques d'indexation utilisées se basent sur l'apparence de l'image.

Dans la première technique d'indexation, les emplacements des objets sont bien identifiés dans l'image. La caractéristique de la forme des objets est calculée à partir de l'opérateur de Marc/Hildreth [Fadi 98], l'index est appelé MH_index (MH : Marc Heldreth).

Dans la seconde technique, la caractéristique visuelle est basée sur des images intermédiaires obtenues à partir de la transformée de Fourier de l'image d'origine composée d'une scène complexe, cet index est noté TF_index (TF : Transformée de Fourier).

Un fichier est généré pour chaque index utilisé et est constitué de plusieurs lignes, une pour chaque image. La structure d'une ligne est la suivante :

<La référence de l'image > <le vecteur permettant de l'identifier>

Chaque vecteur est constitué de vingt réels qui vont permettre de représenter l'image selon la caractéristique visuelle voulue.

Le moteur de recherche

Le moteur est composé de deux engins de recherche l'un spécialisé dans la recherche des images et l'autre dans la recherche de documents textuels. Comme nous l'avons déjà mentionné, les requêtes traitées dans notre système sont conformes au standard Z39.50. Lorsque celles-ci sont reçues par le serveur Z39.50, elles sont analysées et si elles ne sont pas conformes au standard alors elles sont rejetées. Une fois l'étape d'analyse franchie, elles sont transmises au moteur de recherche qui les décompose en deux requêtes, la première est transmise au moteur de recherche textuelle et la seconde au moteur de recherche basée sur le contenu comme l'indique la Figure 4.7. A partir de cette étape, le moteur de recherche a le choix d'appliquer l'une ou l'autre des stratégies décrites dans section planification du moteur de recherche.

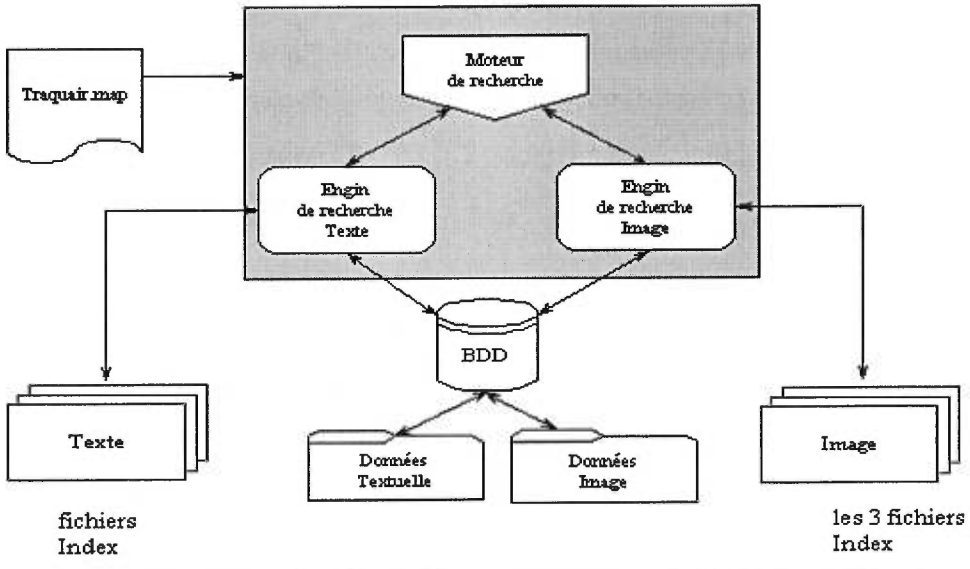


Figure : 4.7 Structure du moteur de recherche.

L'engin de recherche textuelle

Cet engin se base sur la fonction Isearch proposée par Isite et utilise les correspondances entre les métadonnées associées à un document et les attributs bib1 du standard Z39.50 stockés dans le fichier map.ini décrit dans la section suivante.

Le fichier map.ini

Ce fichier fait la correspondance entre les attributs définis pour la base de données Traquair et ceux du standard Z39.50. Ainsi, à la réception d'une requête au standard Z39.50, le moteur de recherche récupère dans ce fichier la correspondance entre les références de bib1 et les champs de la base de données pour pouvoir faire la recherche.

Traquair]
bib1/4=TITLE
bib1/20=COLLECT
bib1/30=PICDATE
bib1/32=BUILDTDATE
bib1/45=MJTYPE
bib1/46=MNTYPE
bib1/47=PHTYPE
bib1/58=STREET
bib1/59=CITY
bib1/60=PROVINCE
bib1/1003=PHOTOGRAPHER
bib1/1010=DESCRIPTION
bib1/1023=INDEX
bib1/1024=DIM
bib1/1028=SIZE
bib1/1032=IREF
bib1/1034=TYPE

Dans cette liste, les éléments gauches des équations sont des attributs standards de Z39.50 dans l'ensemble bib1 et les éléments à droite sont les attributs correspondants dans la base Traquair.

Dans notre cas, nous n'utilisons qu'une seule base de données appelée Traquair. Une nouvelle base de données peut être ajoutée simplement en indiquant dans ce fichier les correspondances voulues comme suit :

[Livre]

bib1/4=Titre

bib1/1003=AUTEUR

bib1/1010=SUJET

Un exemple de transformation de requête Z39.50 est donné ci-dessous

Notman[1,1003]

qui permet de rechercher les documents images du photographe Notman. Comme 1003 est associé à PHOTOGRAPHER dans le fichier Traquair-map.ini, la requête est transformée :

PHOTOGRAPHER / Notman

Les paramètres de la commande Isearch sont les suivantes:

Isearch	[-d (X)]	// Chercher dans la base de données X.
	[-p (X)]	// Créer l'ensemble X des résultats.
	[-q]	// Afficher les résultats et quitter.
	[-and]	// Faire un Et logique sur les résultats.
	[-rpn]	// Requête en RPN.
	[-infix]	// Interpréter la requête comme une requête algébrique.

Exemple: Isearch -d HISTOIRE -rpn titre/chat titre/chien or titre/souris and

Cette requête permet de rechercher dans la base de données HISTOIRE les documents dont le titre contient le mot chat, chien ou souris.

Ce moteur génère en sortie le résultat de la recherche dans un fichier en indiquant, pour chaque document retrouvé, sa référence et un facteur de similarité. Nous avons aussi un ensemble de champs à afficher comme par exemple, l'auteur du document et la description résumée du document. La structure du fichier de résultats est la suivante:

< Référence du fichier > : Ref1	*// Référence du document trouvé*
< Similarité > : Sim1	*// Facteur de similarité*
<Auteur> : xxx	*// Nom de l'auteur*
<Description> :	*// Description du document*
##	// Séparateur entre les documents

L'engin de recherche image

L'engin de recherche développé reçoit en entrée une requête au format Z39.50. Des transformations lui sont appliqué en utilisant les correspondances décrites dans le fichier map.ini. Après la transformation, une recherche est effectuée en se basant sur le programme IsearchImage [Fadi 98]. Ce programme reçoit en entrée l'image exemple utilisée pour la recherche et le type d'index à utiliser. Il crée d'abord l'index de cette image, ensuite calcule la similarité de cet index avec ceux des images de la base. Le pseudo-algorithme utilisé est le suivant :

Recherche_Image (Image_recherchée, Type_Index)
Début
Ouvrir le fichier de type Type_index *// fichier index de la base image*
Pour chaque image dans le fichier *// une ligne : ref image vecteur*
Début
Calculer la distance de image_exemple à l'image courante
Mettre dans un fichier résultat le nom de l'image et la distance calculée
Fin
Trier le fichier résultat selon la distance (de la plus proche à la plus éloignée)
Fin

La distance entre deux images est la distance euclidienne entre les vecteurs associés aux images.

Soit V1 et V2 les vecteurs associés respectivement aux images M1 M2.

$V\ 1 = (x1, x2, \ldots x20),$

$V\ 2 = (y1, y2, \ldots y20);$

La distance entre les images M1 et M2 est donnée par :

D (M1 , M2) = Distance Euclidienne (M1,M2)

Le résultat obtenu est un fichier contenant les images et leur distance à l'image exemple. Ces distances sont normalisées.

Regroupement des résultats

Les résultats des deux recherches décrites précédemment sont combinées en un seul fichier. Celui-ci contiendra la référence de l'image trouvée et sa similarité. Chaque moteur de recherche retourne une liste d'images, nous faisons l'intersection des deux listes pour retrouver le résultat final à retourner à l'usager. La similarité finale du résultat est calculée en utilisant les similarité retournée par les moteurs texte et image. En effet, si une image est retrouvée par les deux moteurs alors la similarité finale est la somme des similarité, texte et image.

4.4.4 Utilisation de l'API Z39.50

L'interaction entre les moteurs de recherche et le serveur Z39.50 n'est possible que si des règles permettant la communication des données, requêtes et résultats, sont respectées. Nous décrivons l'API qui a été utilisée pour assurer ce mécanisme de communication.

La requête reçue par le serveur Z39.50 est transmise au moteur de recherche. Dans le fichier sapi.ini sont déclarés les chemins d'accès à deux scripts, le premier est le script de recherche et il correspond au moteur de recherche, le second est le script qui permet de lire un document en entier. La Figure suivante résume le mécanisme d'API utilisé pour les échanges entre le serveur Z39.50 et le moteur de recherche dans les deux sens, c'est à dire pour l'envoi des requêtes et pour la lecture des résultats.

Comme on le voit sur la Figure 4.8 ci-dessus, le serveur utilise le fichier Zserver.ini pour savoir où se trouve le fichier sapi.ini pour la base de données demandée dans la requête. Dans ce fichier est inscrit le chemin d'accès au script de recherche de documents. Le processus fils créé par le serveur lance un processus de recherche en lui indiquant la requête et le nom du fichier où aller mettre le résultat. Ce nom de fichier est constitué du numéro de processus fils. A la fin de création du fichier de résultats, celui-ci est retourné au serveur qui s'occupe de le restituer au client Z39.50.

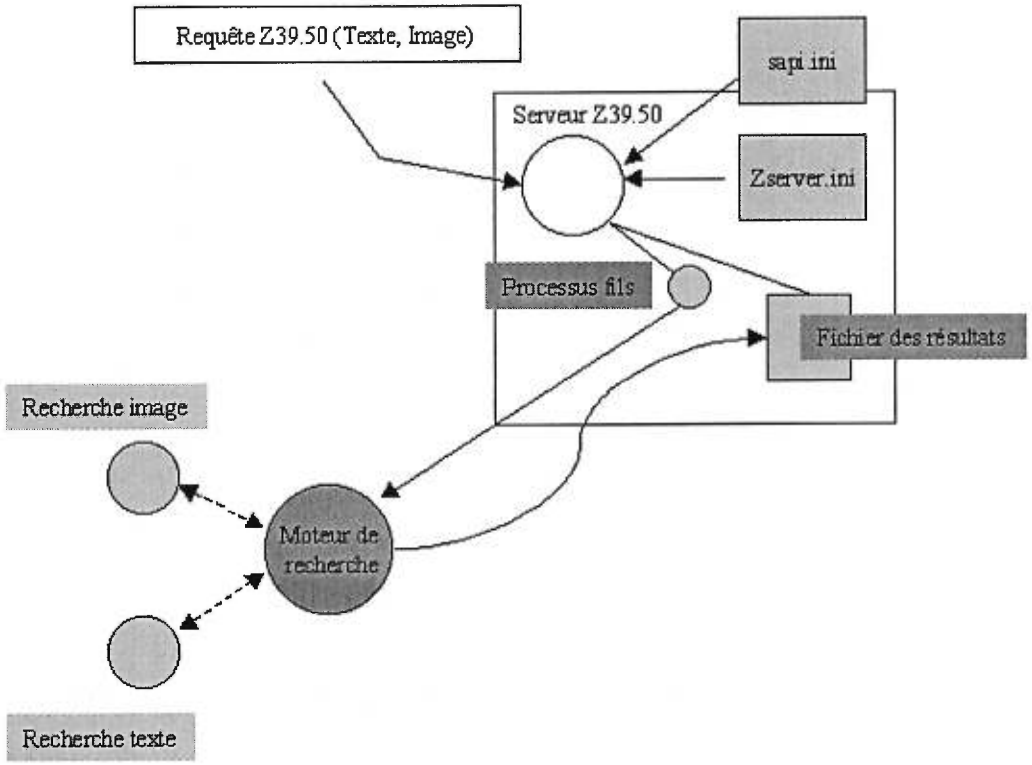


Figure 4.8 : Utilisation de l'API pour transmettre la requête et récupérer la liste de résultats.

C'est approximativement le même mécanisme qui est utilisé pour pouvoir récupérer le contenu d'un document. Dans le fichier sapi.ini est spécifié le chemin d'accès à un script qui permet de lire un document. Le processus fils créé par le serveur Z39.50 lance le script avec la référence du document à récupérer et indique où aller mettre le résultat.

4.5 Le Système d'Interface Graphique

L'interface au système est le moyen que les usagers utilisent pour composer et soumettre des requêtes d'interrogation. Elle permet de rechercher des documents de type multimédia composés de textes et d'images, et présente les caractéristiques principales suivantes :

- utilisable à travers un fureteur,
- permet la composition de requêtes basées sur le texte,
- permet la composition de requêtes basées sur le contenu pour la recherche d'images,
- permet d'afficher les résultats sous la forme d'une liste accompagnée d'une description sommaire de chaque document trouvé,
- permet la navigation dans la liste des résultats trouvés.

Deux aspects sont à distinguer dans l'interface graphique, la partie des options qui accompagnent la requête et la requête elle même. Les options de recherche qu'un usager peut préciser sont:

- la base de données à utiliser pour la recherche,
- le type de documents recherchés,
- le domaine de recherche,
- la langue dans laquelle le document est écrit.

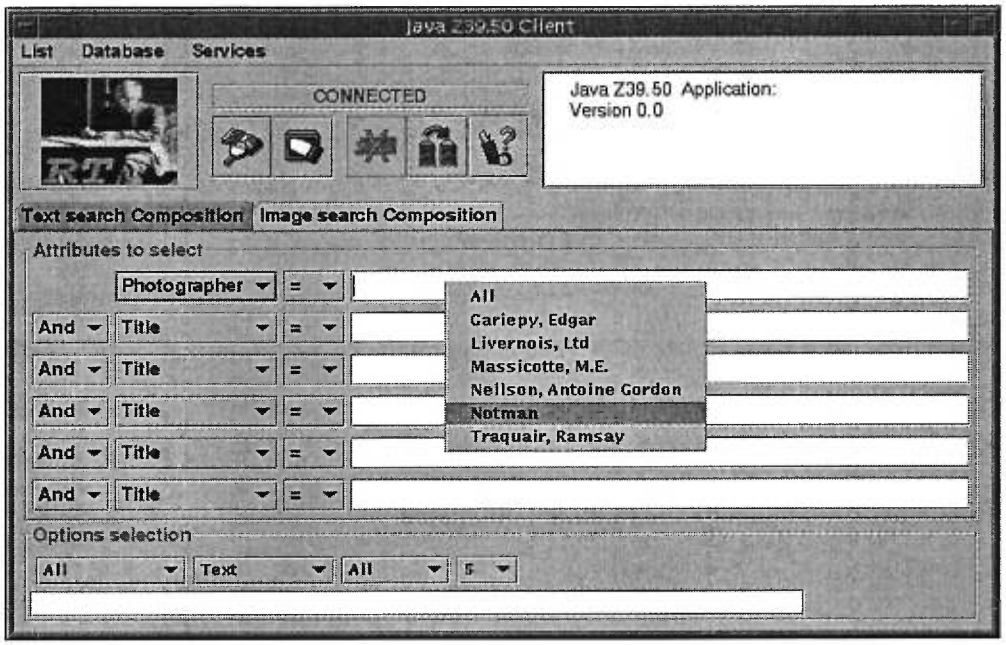


Figure 4.9 : Interface graphique utilisée pour les requêtes textuelles.

Deux modes de spécification de requêtes sont possibles, le premier est utilisé pour le texte et le second pour l'image. Une requête peut contenir les deux spécifications qui doivent être satisfaites par les documents retrouvés (un ET logique implicite). Une requête de type textuelle combine plusieurs attributs à l'aide d'opérateurs logiques. Les champs de recherche proposés sont par exemple un nom d'auteur, un titre, un sujet ou un texte libre (Voir Figure 4.9).

L'usager peut utiliser un service de traduction de l'anglais au français pour s'aider dans la composition de requêtes avant leur envoi au système de recherche.

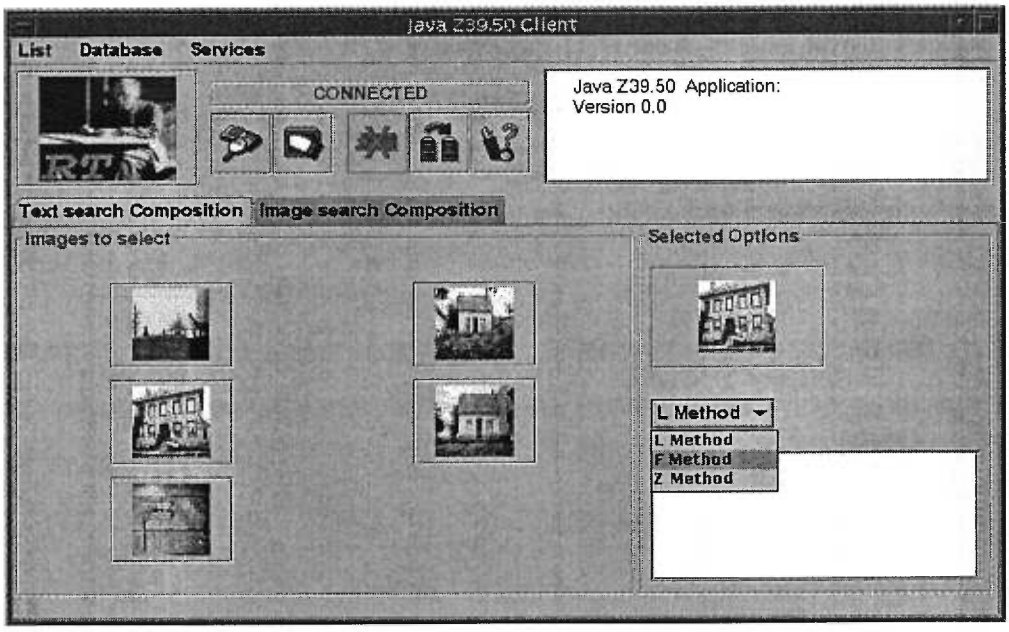


Figure 4.10 : Interface graphique utilisée pour les requêtes visuelles.

Une requête de type image est constituée d'une image de référence choisie parmi un groupe d'images et l'un des trois types d'indexation image développés à McGill [Fadi 98] et qui sont (voir Figure 4.10):

- *IA_index*,
- MH_index.,
- TF_index.

Pour le moment, nous avons mis un petit nombre d'images exemples dans l'interface. Cependant, le nombre peut être facilement augmenté et les images exemples peuvent aussi être organisées hiérarchiquement pour faciliter la sélection par l'usager. Ce sont des extensions faciles à implanter dans le futur.

Le premier type d'index est très simple et ne permet pas d'avoir de bons résultats de recherche. Dans le deuxième cas, les objets d'une image sont bien définis et l'arrière plan est simple, de meilleurs résultats sont obtenus dans ce cas. Concernant le troisième type d'index, il est destiné à des images dont la scène est complexe; cette technique donne de meilleurs résultats.

Les résultats sont affichés dans un troisième panel sous la forme de liens URL avec un résumé du document. Si l'usager clique sur le lien alors tout le document est rapatrié via le serveur Web pour être affiché au niveau du fureteur à partir duquel l'interface s'est lancée (voir Figure 4.11).

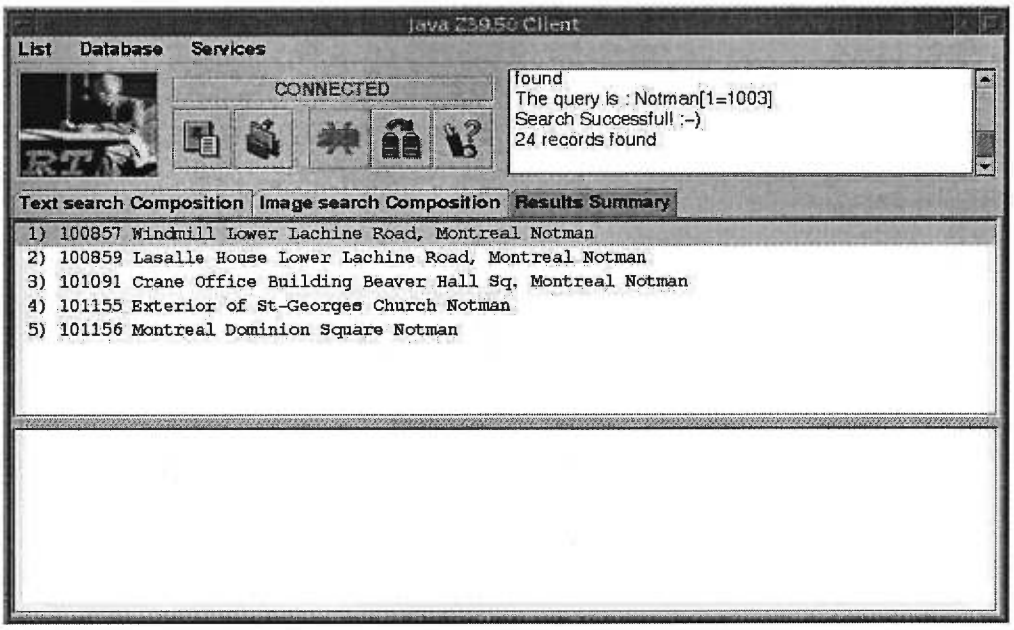


Figure 4.11 : Interface de récupération des résultats. L'usager peut choisir un document à visionner dans la liste.

Le choix d'un document de la liste de résultats permet d'afficher l'image et toutes les informations textuelles qui l'accompagnent (voir Figure 4.12 page suivante).

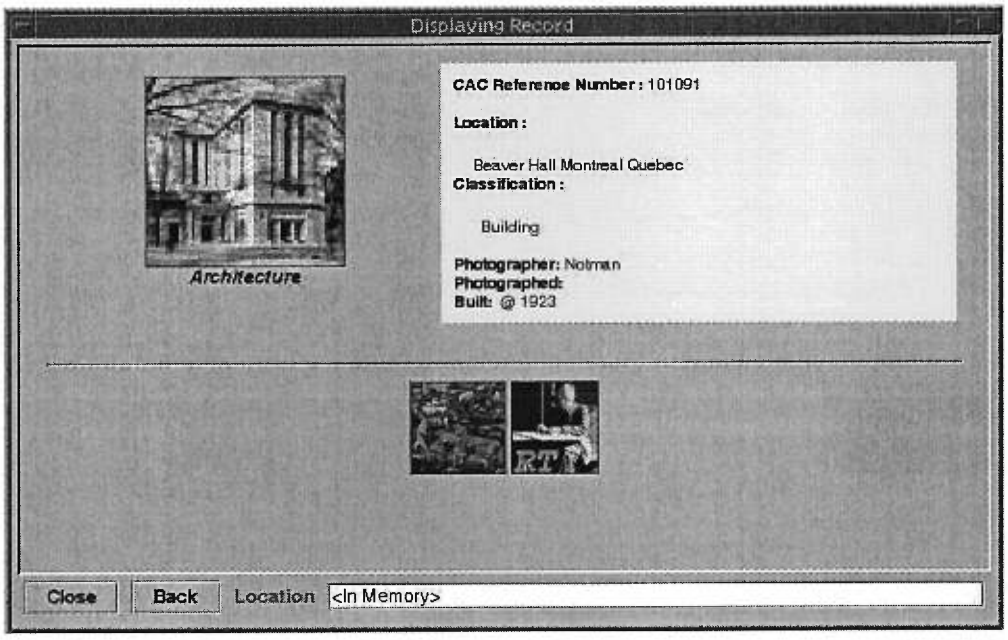


Figure 4.12 : Un document sélectionné par un double click.

4.6 Conclusion

Dans ce chapitre, nous avons expliqué les techniques qui ont été utilisées pour la mise en œuvre du prototype. Nous nous sommes basés sur le système Isite pour la partie standard d'interrogation et sur une API java pour la conception du client Z39.50. Un système d'API a été utilisé pour la transmission des requêtes par le serveur au moteur de recherche et la récupération des résultats.

Le premier avantage que nous avons eu à utiliser le standard est de pouvoir accéder de manière uniforme à différentes bases de données. Le mécanisme d'interface entre le serveur Z39.50 et les applications propres aux bases de données est performant et très pratique.

Concernant l'implémentation de la plate-forme d'agents, nous avons utilisé un ORB qui nous a permis de faire abstraction des problèmes de communication. Nous nous sommes concentrés sur l'aspect fonctionnalité du système. L'utilisation du système HORB nous a permis de voir de plus près les problèmes de programmation distribuée et d'élaborer une solution à notre problème.

Chapitre 5

5 Conclusion

Ce rapport présente les résultats de recherche d'une partie du projet mené conjointement par plusieurs universités pour l'étude des systèmes de bibliothèque numérique multimédia. Ce projet a consisté à étudier et implanter un prototype de bibliothèque numérique.

Les points importants abordés lors de cette étude sont:

- La définition de l'architecture globale du système de bibliothèque numérique,
- L'organisation et l'accès aux données,
- La mise en œuvre des moyens d'interrogation des données.

L'étude des systèmes existants nous a permis d'identifier les principales caractéristiques que devrait présenter notre prototype. Nous avons passé en revue les architectures des systèmes Harvest pour l'utilisation de la notion de services et UMDL pour l'aspect multi-agents. Nous avons constaté des lacunes quant à l'organisation des données et aux possibilités d'extensibilité. Basé sur les résultats de la synthèse des systèmes étudiés, nous nous sommes fixés un certain nombre d'objectifs à atteindre pour la réalisation du prototype de BND:

- Gérer une masse considérable de données,
- Offrir un accès rapide aux données,
- Assurer l'évolution des données du système.

Pour atteindre nos objectifs, nous avons mis en œuvre un système de gestion de données qui utilise un catalogue à deux niveaux, un système de services basé sur une hiérarchie d'agents et mis en place un système d'interrogation basé sur une variante du standard Z39.50 pour la recherche d'images.

La bibliothèque numérique, qui utilise plusieurs bases de données réparties sur différents sites, s'organise autour d'un catalogue de données à deux niveaux. Le premier niveau regroupe les méta-données décrivant les différentes bases de données et le second est constitué par les catalogues propres aux bases de données que le système va desservir. Les avantages de la hiérarchisation des catalogues sont les suivants:

- Traiter rapidement les requêtes de recherche en les dirigeant, dès leur soumission, vers les sites potentiels,
- Eviter la surcharge du réseau en évitant d'envoyer les requêtes vers les sites inadéquats,
- Prendre en compte de nouvelles bases de données en déclarant les nouveaux sites.

Parmi les inconvénients de cette solution, on peut citer la vulnérabilité aux pannes et la création d'un point de contrôle au niveau du site où se trouve le catalogue global. En effet, un goulot d'étranglement peut se former lorsque l'affluence des requêtes est très élevée. Une solution à ce problème serait de prévoir plusieurs sites miroirs contenant le catalogue global vers lesquels seront redirigées les requêtes en cas de surcharge.

Différents types d'agents ont été utilisés pour mettre en place le système. Certains d'entre eux sont indispensables au fonctionnement du système et constituent les éléments de base, comme par exemple, l'agent superviseur, les agents de recherche globaux et les agents de recherche locaux. Pour chaque nouveau service introduit dans le système, nous utilisons un ou plusieurs agents pour l'implémenter. Cette architecture

multi-agents nous a permis de faire face aux problèmes de traitement rapide des requêtes ainsi que la gestion de l'extension en l'enrichissant par de nouveaux services ainsi que de nouvelles bases de données. Comparés aux agents utilisés dans les systèmes sophistiqués comme UMDL, les agents que nous avons utilisés sont simples et faciles à implémenter. Néanmoins, des services plus complexes et plus riches peuvent être conçus et intégrés au système.

L'organisation des agents en niveaux hiérarchiques présente l'avantage d'avoir un système extensible facile à contrôler. L'inconvénient de cette approche est la prolifération des agents au fur et à mesure que l'extension du système devient importante. En effet, leur gestion par un seul agent superviseur rendrait le système très lourd d'où la nécessité d'introduire d'autres niveaux de hiérarchie. Cette hiérarchie permettra de constituer des niveaux entre les agents de supervision et d'attribuer à chaque superviseur un ensemble de sous-agents superviseurs.

Quant à l'interopérabilité des différents sites, elle a été assurée par l'utilisation d'un standard d'interrogation qui initialement a été proposé pour la recherche documentaire. La nouveauté introduite dans notre projet de recherche est la proposition de l'extension de ce standard. Cette extension se manifeste par la possibilité de faire des recherches d'images en se basant sur des caractéristiques visuelles comme la luminosité et la couleur. Ces caractéristiques visuelles peuvent être combinées avec des informations textuelles pour affiner la recherche. Nous avons respecté la philosophie selon laquelle le standard est organisé en proposant un nouvel ensemble d'attributs d'interrogation de documents images et les index images à utiliser. Les modifications apportées au standard nous ont permis de proposer deux types de recherche dans le prototype, la recherche de documents textuels et la recherche de documents images.

L'implantation du système a été essentiellement réalisée en langage Java. Les principaux points qui ont dominé l'implantation sont : l'aspect réparti de l'application, la conception du client Z39.50 et l'interface d'interrogation graphique.

Plusieurs choix se présentaient à nous pour l'implantation de l'aspect réparti de l'application : Voyager, RMI et HORB. Pour le premier système, nous avons bien sur constaté sa richesse en fonctionnalités; cependant, dès le début des tests, nous nous sommes rendus compte d'un certain nombre de problèmes pour la création d'agents ou d'objets à distance, problèmes qui ont été résolus dans la version suivante de Voyager. RMI a été exclu à cause de la charge de travail qui s'imposait pour réaliser des fonctionnalités qui étaient déjà présentes dans le système HORB, d'où notre choix. Il est intéressant d'explorer l'utilisation d'un système plus performant comme « Voyager » ou « Corba » pour la mise en œuvre d'un prototype expérimental riche en fonctionnalités et présentant de meilleurs performances.

En ce qui concerne l'implantation du client Java, le respect rigoureux des spécifications du standard et l'utilisation de classes proposées par des groupes de recherche nous ont facilité grandement la tâche. La seule difficulté a été au niveau de l'extension. En effet, nous nous sommes basés sur un serveur Z39.50 existant, donc respectant le standard Z39.50. Nous avons été obligés d'utiliser des codes existants. La signification de ces codes a été modifiée pour répondre aux exigences de la recherche d'images.

Cette étude nous a permis de prouver la faisabilité d'un système de bibliothèques numériques par la construction d'un prototype fonctionnel. Nous avons aussi constaté les difficultés auxquelles nous avons à faire face pour rechercher des documents multimédia, dans notre cas, les images. En effet, la complexité s'accroît en voulant utiliser des méthodes de recherche plus complexes comme par exemple la forme des objets dans l'image, la position d'un objet par rapport à un autre, etc., des cas pour lesquels l'utilisation de langages d'interrogation spécialisés s'impose.

Dans cette étude, nous nous sommes imposés certaines limites pour simplifier la réalisation. Cependant, Plusieurs axes restent à explorer, nous pouvons citer la création d'agents plus sophistiqués et leur intégration au système comme par exemple un agent de gestion de profil usager ou un agent d'interprétation de requêtes pour faire de l'expansion automatique.

Cette étude a été sanctionnée par la publication d'un article [Hamard 99] dans lequel nous avons fait part des avantages qu'offre notre architecture ainsi que les problèmes qui restent encore posés.

6 Références bibliographiques

[Andressen 95] D. Andresen & al.
The WWW Prototype of Alexandria Digital Library.
http://www.dl.ulis.ac.jp/ISDL95/proceedings/pages75/17.html

[Birmingham 95] William P. Birmingham
The University of Michigan
Electrical Engineering and Computer Science Department
School of Information Science and Library Studies
Ann Arbor, Mi 48109
D-Lib Magazine, July 1995.
http://www.dlib.org/dlib/July95/07birmingham.html

[Bowman 94] Udi Manber and Michael F.
The Harvest Information Discovery and Access System.
C. Mic Bowman, Peter B. Danzing Darren R. Hardy,
CU-CS-732-94,
Department of Computer Science, University of Colorado.

[Chang 97] Shih-Fu Chang & al.
Finding Images / Video in Large Archives.
D-lib Magazine, February 1997.
http://mirrored.ukoln.ac.uk/lis-journals/dlib/dlib/dlib/february97/columbia/02chang.html

[Corey 94] Corey James F
1994 "A grant for Z39.50." Library hi tech. 12(1):37-47.

[CNIDR] The Isite Information System version 1.04
The Clearinghousing for Networked Information Discovery and Retreival.

[Dublin Core] http://linnea.helsinki.fi/meta/oclc/index.htm

[Duda,94] Andrzej Duda and Mark A. Sheldon
14th IEEE International Conference on Distributed Computing Systems,
Poznan, Poland, June 1994.
http://www.psrg.lcs.mit.edu/publications/Papers/icdcsabs.htm

[Fadi 98] Fadi Beyrouti
Content Based Image Retrieval and Scene Classification by Appearance.
Departement of Electrical Engineering
McGill University, September 1998.

[Grossman 95] Grossman R., Qin X. and Xu W.
An Architecture for a Scalable, High-Performance Digital Library.

http://www.computer.org/conferen/mss95/hulen/hulen.htm

[Gray 82] R.M. Gray et Y. Linde : "Vector quantizers for Gaus-Markov sources" ;
IEEE Trans. On Comm. Vol. COM-30, No. 2,
Fevrier 1982, pages. 381-389.

[Hylton 94] Hylton J.
Library 2000 Architecture
http://ltt-www.lcs.mit.edu/ltt-www/Public/Architecture/index.html

[Hirano 96] Hirano Satoshi
The Image Carpet for Network Computing: HORB Flyer's Guide.
1996/03/21 for HORB 1.2
http://openlab.etl.go.jp/horb/

[Hamard 99] K. Hamard, J.Y. Nie G. Bochmann B. Kerhervé et al.
A System of Digital Library Based on Multi-level Agents.
First workshop on Agent Oriented Information Systems (AOIS'99)
May 1999, Seattle. pages 77-91.

[Jennings 96] N. R. Jennings et M. Wooldridge, « Software agents »
IEEE Review, pp 17-20, 1996.

[Kacmar 94] Charles Kacmar et al.
An Architecture and Operation Model for a Spatial Digital Library.
D-Lib Magazine 1994.
http://abgen.tamu.edu/DL94/paper/kacmar.html

[Mechkour 95] Mechkour M.
EMIR2. Un modèle étendu de représentation et de correspondance d'images pour la recherche d'informations. Application à un corpus d'images historiques.
Thèse de doctorat.
Université Joseph Fourier – Grenoble I. Novembre 1995.

[Moen 95] William Moen.
The ANSI/NISO Z39.50 Protocol: Information Retrieval
in the Information Infrastructure
http://www.cni.org/pub/NISO/docs/Z39.50-brochure/50.brochure.toc.html

[Myron 95] Myron Flickner, Harpreet Sawhney
Query by Image and Video Content: The QBIC System
Computer, Vol. 28, No. 9, September 1995
http://computer.org/computer/co1995/r9023abs.htm

[Ogle 96] Virginia Ogle and Robert Wilensky.
Testbed Development for the Berkely Digital Library Project.
D-Lib Magazine July/August 1996.
http://mirrored.ukoln.ac.uk/lisjournals/dlib/dlib/dlib/july96/berkeley/

07ogle.html

[Paepcke 96a] Paepcke A.. & al.
Using Distributed Objects for Digital Library Interoperability.
IEEE computer Society.May 1996. P 61-68.

[Paepcke 96b] Andreas P. & al.
Toward Interoperability in Digital Libraries Overview and Selected Highlights of Stanford Digital Library Project.
http://elib.stanford.edu/Dienst/UI/2.0/Describe/stanford.cs

[Salton 85] G.Salton, M. J. McGill
Introduction to modern information retrieval.
McGraw-Hill, NewYork 1985.

[Shivakumar 95] N. Shivakumar and H. Garcia-Molina,
SCAM: Acopy detection mechanism for digital document,
Digital Library'95, 1995,
http://csdl.tamu.edu/csdl/DL95/papers/shivakumar.ps

[Swain 91] Swain M. J. et Ballard D. H.,1991. «Color Indexing »,
International Journal of Computer Vision, vol. 7, No.1, p. 11-32.

[Terence 96] Terence R. Smith
The Meta-Information Environment of Digital Librairies
D-lib Magazine July/August 1996
http://www.dlib.org/dlib/july96/new/07smith.html

[Van Risjbergen 79] C.J. Van RIJSBERGEN
Information Retrieval.
Second Edition, Buttersworth, London, 1979.

[William 95] William Y. Arms.
Key Concepts in the Architecture of Digital Library.
D-Lib Magazine, July 1995.
http://www.dlib.org/dlib/July95/07arms.html

[Wooldridge 94] M. Wooldridge & N.R. Jennings
«Agent Theories, Architectures, and Languages: A Survey »
Procedings of. ECAI-Workshop on Agent Theories, Architectures and Languages, M. J. Wooldridge and N. R. Jennings, Amsterdam, Pays-Bas, 1994

[Youngchoon 97] Yougchoon Park & Forouzan Glshani
ImageRoadMap: A New Content-Based Image Retrieval System
Lecture Notes in Computer Science 1308.
Database and Expert Systems Applications
8th International Conference, DEXA'97
pp 225-239.

[Yapo 98] Annick Yapo, 1998.
Modélisation des méta-informations pour documents visuels.
Mémoire de recherche, maîtrise en informatique de gestion.
Université du Québec à Montréal.

[ZIG 97] Attribute Set BIB1 (Z39.50-1995) : Semantics. October 1997.

Annexe

A

7 Description du Standard Z39.50

A.1 Les Eléments de Base du Standard

Contrairement aux protocoles Internet comme HTTP et WAIS, Z39.50 est un protocole orienté session. C'est à dire qu'une session persistante est démarrée lorsqu'une connexion au serveur Z39.50 est effectuée. Dans ce cas, une connexion est maintenue tant que la session reste ouverte [Corey 94].

Les protocoles orientés session sont plus performants que les protocoles orientés transaction qui demandent qu'une connexion soit établie pour chaque envoi de message, sans négliger le fait que c'est le moyen qui permet de faire le raffinement itératif du résultat de recherche. C'est aussi le moyen qui permet au serveur et au client de négocier les services à maintenir tout au long de la session. Ce n'est pas le cas du protocole HTTP pour lequel les messages échangés doivent contenir la description de certaines caractéristiques que le serveur doit avoir et ceci pour chaque transaction.

Dans sa forme la plus simple, le standard Z39.50 est un protocole synchrone. Dans ce cas, le client envoi un message au serveur et attend la réponse [Moen 95].

Le protocole spécifie les formats et les règles gouvernant les échanges entre un client et un serveur. Ces échanges permettent au client de demander au serveur de chercher dans

une base de données et d'identifier les enregistrements qui vérifient des critères spécifiés et d'accéder à l'ensemble ou à une partie des enregistrements trouvés.
Ce protocole permet l'interrogation simultanée de banques de données par l'envoi d'une requête unique à *plusieurs serveurs distants* sans se préoccuper de l'interface propre au module de recherche utilisé par les usagers directs de ces banques de données.

Il permet aussi de sélectionner, dans la marée de serveurs existant sur Internet, le ou les serveur(s), et à l'intérieur de ces serveurs la ou les base(s) de données à qui adresser une requête, sans se limiter aux seuls documents numériques accessibles directement à travers l'Internet, mais au contraire en étendant la recherche aux bases de données de références bibliographiques.

Le standard Z39.50 et les *services* qu'il fournit prévoient non seulement la navigation dans les bases de données selon le mode hypertexte, mais également le parcours de listes d'indexes sur un serveur, ou mieux encore, la pratique du *"relevance feedback"*, procédé qui réutilise un paquet de références reçues en réponse à une requête, ou un sous-ensemble préalablement sélectionné, ou même une seule référence, pour relancer une recherche des documents s'en rapprochant le plus, surmontant ainsi le problème que posent, dès lors qu'on interroge plus d'une base, les différences de vocabulaire d'indexation.

Il est implémenté au-dessus du protocole TCP/IP et permet à tout client Z39.50 d'envoyer une requête de recherche indiquant une ou plusieurs bases de données et les paramètres qui permettent de préciser si les réponses doivent inclure ou non les enregistrements trouvés dans la réponse. Le serveur répond par le nombre d'enregistrements trouvés. La responsabilité de la récupération des enregistrements est laissée à la charge du client. En effet, le serveur organise les enregistrements trouvés en ensembles et ainsi il permet au client de récupérer ceux qui l'intéressent grâce à leurs positions. Les possibilités de base qui doivent être offertes sont les suivantes :

- Le client doit pouvoir indiquer un ensemble de paramètres lui permettant de préciser les données de l'enregistrement à récupérer. Exemple, le client peut demander au serveur de retourner l'ensemble des enregistrements trouvés si leur nombre ne dépasse pas cinq; sinon, une description sommaire de chaque enregistrement est récupérée.

- Le client peut indiquer une syntaxe pour les réponses, selon le format texte ou USMARC par exemple.
- Le client peut nommer l'ensemble des enregistrements retrouvés pour les référencer dans des transactions ultérieures.
- Il peut détruire cet ensemble d'enregistrements.
- Le serveur peut imposer un accès réduit en demandant une authentification avant toute opération de recherche.

A.2 Les Grammaires de Requêtes

Z39.50 définit plusieurs grammaires de requête chacune identifiée par un numéro. Les requêtes de type-0 sont utilisées pour les grammaires privées. Parfois, un organisme préfère utiliser sa propre grammaire dans l'implémentation de son client et son serveur. Les requêtes de type-1 sont les requêtes les plus largement utilisées. Se sont des requêtes en notation polonaise inverse (RPN). La structure d'une requête est la suivante :

RPN-Query : : = Argument | Argument + Argument + Operator
Argument : : = Operand | RPN-Query
Operand :: = AttributeList + Term | ResultField | Restriction
Restriction :: = ResultSetId + AttributeList
Operator :: = AND | OR | AND-NOT | Prox
ResultSetId :: = Identification / pointeurs sur l'ensemble des enregistrements trouvés
Term :: = Un mot, une phrase, un ensemble de mots, un nombre, ou toute autre information à rechercher.

Exemple 1: Rechercher tous les enregistrements qui contiennent dans l'attribut titre un mot qui commence par « educ ».

La requête :

RPN-Query : :=

AttributeList =

(use, title),

(relation, equal),

(structure, word),

(truncation, right),

(completeness, incomplete subfield),

(position, any),

Term = educ

Les requêtes de type-2 utilisent la grammaire de l'ISO 8777. Elle souffre de certaines limites. Cependant, ces requêtes peuvent être envoyées comme des requêtes de type-0.

Les requêtes de type-100 utilisent la grammaire du standard ANSI/NISO CCL(Common Command Language). Cette grammaire souffre des même limites que les requêtes de type-2.

Les requêtes de type-101 sont une extension des requêtes de type-1. Elles permettent de supporter des recherches par proximité. Avec la version 3 du standard Z39.50, les requêtes de type-101 sont identiques à celles du type-1, cependant, elles sont différentes dans la version 2 du standard. [Moen 95].

Formulation de requêtes

Les requêtes formulées sont constituées de termes et à chacun d'eux est associé un ensemble d'attributs. Ces attributs permettront d'indiquer les caractéristiques associées à chacun des termes de la requête. Le serveur est responsable de l'association des attributs à la conception logique de la base de données. Les termes peuvent être combinés dans des requêtes booléennes exprimées dans la notation polonaise inverse [ZIG,97].

Il existe six types d'attributs :

- les attributs d'utilisation (Use attributes),
- les attributs de relation (Relation attributes),
- les attributs de position (Position attributes),
- les attributs de structure (Structure attributes),
- les attributs de troncature (Truncation attributes),
- les attributs de complétude (Completeness attributes).

Les *attributs d'utilisation (1)* constituent un point d'accès (nom, titre, sujet, ...). A titre d'exemple, nous pouvons citer l'ensemble d'attributs BIB-1. Chacun des attributs est identifié par un numéro unique qui sera utilisé de manière standard dans une requête. Dans la table suivante nous donnons un extrait de la table de BIB-1.

Table A1 : Les attributs d'utilisation

Use	**Value**	**Reference to group name**
Personal name	1	Name-personal
Corporate name	2	Name-corporate
Title	3	Title
...	...	...
Author	1003	Author
Any	1015	Any

Les *attributs de relation (2)* décrivent les relations entre les points d'accès (partie gauche d'une relation) et les termes utilisés dans la recherche (partie droite de la relation). Exemple,

Date-publication <= 1975

Les attributs de relation sont les suivants :

Table A2 : les attributs de relation

Ralation	**Valeur**
Less than	1
Less than or equal	2
Equal	3
Greater than	4
Greater or equal	5
Not equal	6
...	

Les *attributs de position (3)* spécifient la position des termes de recherche à l'intérieur des champs dans lesquels ils apparaissent. Les attributs de position sont les suivants :

Table A3 : Les attributs de position

Position	**Valeur**	**Définition**
First in field	1	Rechercher le terme qui apparaît le premier dans la champ
First in subfield	2	Rechercher le terme qui apparaît dans un sous champ en première position.
Any position in the field	3	Rechercher le terme dans n'importe quelle position

Les *attributs de structure (4)* permettent de spécifier la structure des termes utilisés dans la requête (exemple un mot simple, une phrase, des mots clés...). Nous avons les structures suivantes : La structure de phrase (1), la structure de mot (2) et la structure mots clés (3).

Les *attributs de troncature (5)* permettent de spécifier si un ou plusieurs caractères peuvent être omis dans les termes utilisés dans la requête. Plusieurs types de troncatures sont possibles comme l'indique le tableau suivant.

Table A4 : Les attributs de troncature

Troncature	**Valeur**	**Structure de l'attribut**
Troncature à droite	1	Mot ou phrase, chaîne, liste de mots
Troncature à gauche	2	-- -- -- -- --
Troncature à droite et à gauche	3	-- -- -- -- --

Les *attributs de complétude (6)* permettent de spécifier que les termes d'une requête représentent un sous champ complet ou incomplet ou un champ complet. La complétude indique que des termes additionnels peuvent apparaître dans les champs ou sous champs avec les termes de la requête.

Il y a trois types de complétude. Le sous champ incomplet (1) , le sous champ complet (2) et le champ complet (3). Dans le premier cas, des mots autres que ceux indiqués dans la recherche peuvent apparaître dans les sous champs ou champs dans lesquels les termes apparaissent. Le sous champ complet (2) indique qu'aucun mot autre que ceux dans la recherche ne peut apparaître dans le sous champ. Le troisième cas stipule qu'aucun mot autre que ceux indiqués dans la recherche ne peut apparaître dans le champ dans lequel les termes existent (Attribute set bib1 Z39.50-1995 Semantics) [ZIG,97].

A.3 Les Services de Base du Standard Z39.50

Le service d'initialisation

C'est le premier service à être utilisé pour commencer une session. Le client et le serveur l'utilisent pour négocier les autres services du protocole.

La négociation : le standard supporte un service de négociation simple. Le client propose des valeurs grâce à une primitive **InitRequest.** Les réponses envoyées par le serveur sont obtenues grâce à la primitive **InitResponse.** Parmi les valeurs négociées on distingue :

- La version du protocole à utiliser,
- La taille des messages,
- Le nom usager et le mot de passe,

Le service de recherche

Le standard Z39.50 offre un service de recherche performant et efficace. L'ensemble des enregistrements obtenus en réponse à une recherche est persistant au niveau du serveur durant toute la session d'interrogation. Ainsi, le client n'a pas à rapatrier l'ensemble des réponses localement pour ensuite décider des enregistrements à garder. Il peut choisir de récupérer un sous-ensemble d'enregistrements parmi ceux trouvés. Cet ensemble permettra aussi de faire le raffinement de recherche.

L'ensemble des enregistrements trouvés peut être nommé pour de futures références. Dans ce protocole, le client peut fournir dans la requête le nom de l'ensemble dans lequel faire la recherche.

SearchRequest est la primitive qui permet de faire la recherche. La réponse est obtenue via la primitive **SearchResponse.**

Le service de présentation

Le client peut demander au serveur la restitution d'un ou plusieurs enregistrements. Ceux-ci peuvent être récupérés sous une forme complète ou résumée. Les syntaxes les plus communes sont USMARC et SUTRS. USMARC est la syntaxe des enregistrements utilisés dans les bibliothèques Américaines permettant l'échange des informations de catalogage. SUTRS est un format texte simple inventé par ZIG.

Le client peut à n'importe quel moment effacer les enregistrements non désirés, peut naviguer librement dans la liste des enregistrements qui constitue la réponse.

Le service de terminaison de session

Le client ou le serveur peut arrêter une session à n'importe quel instant en utilisant la primitive **disconnect**. Celle-ci ferme la connexion TCP/IP qui existait entre les deux entités [Moen 95].

A.4 Les Différentes Configurations Possibles

L'accès aux bases de données desservies par des serveurs Z39.50 peut se faire soit en utilisant des clients Z39.50 qui se connectent directement aux serveurs, soit en utilisant un serveur Web et une passerelle ou bien en utilisant un client Java (application / applet)

Le première technique a été prévu dès le départ dans le standard; mais l'avènement du Web du langage Java a permis introduire d'autres techniques plus performantes. Les différentes architectures possibles sont résumées sur la figure suivante.

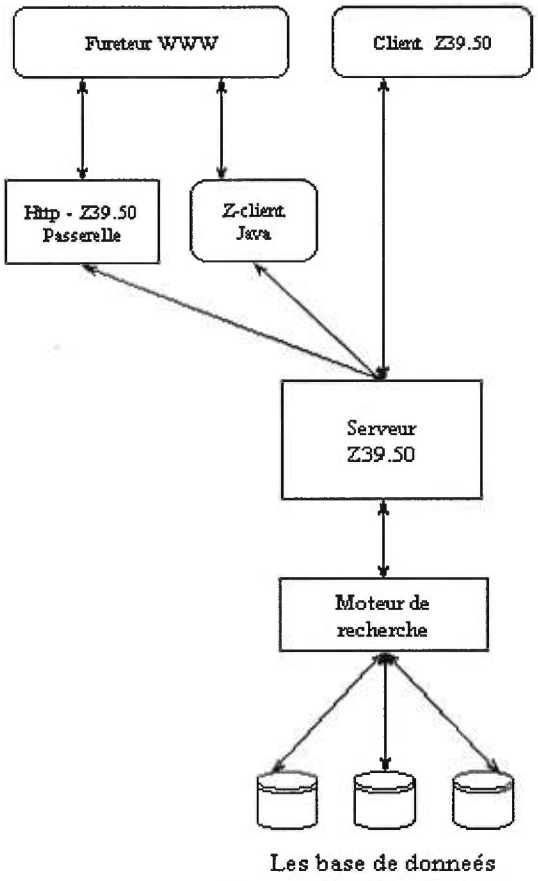


Figure A1 : Les différentes configurations possibles. Client z39.50, Client Java et passerelle Http - Z39.50

Annexe

B

8 Description du Système Isite

Une vue générale de l'architecture du système est présentée par la figure 1. Nous pouvons remarquer que l'interrogation des bases de données peut se faire à travers un client Z39.50 ou à travers un fureteur (Netscape / Internet Explorer) via une passerelle HTTP / Z39.50. Le serveur et l'API de recherche utilisent chacun un fichier de configuration permettant d'initialiser tous les paramètres de fonctionnement.

B.1 Le serveur Zserver

Le serveur (Zserver) offert avec le système Isite implémente les services spécifiés dans la norme ANSI/ISO Z39.50. Il permet à un client Z39.50 de se connecter, de rechercher et d'accéder aux documents retrouvés. Il utilise un fichier de configuration zserver.ini constitué d'un ou plusieurs groupes. Au niveau de chaque groupe sont regroupées des directives.

Le fichier de configuration zserver.ini

Cette liste de directives représente la configuration associée à un serveur particulier. Les directives sont de la forme Attribut / valeur. Le tableau 1 fait la description des directives possibles et spécifie les valeurs utilisées par défaut.

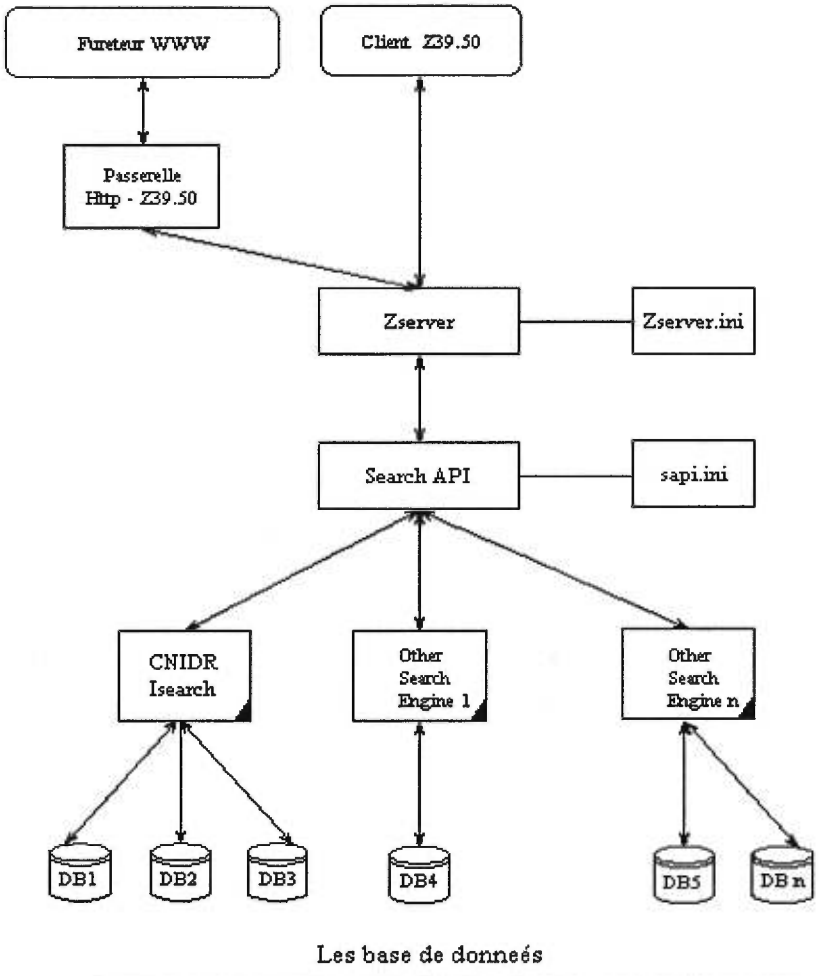


Figure B1 : Architecture du système d'information Isite.

Table B1 : les directives dans le fichier de configuration

Champ	**Valeur par défaut**	**Déscription**
[Nom du groupe]	DEFAULT	
AccessLog= nom de fichier	/tmp/zserver_acess.log	Nom absolu du fichier log
DBList= liste de noms		Les bases de données accessibles
DebugLevel=entier (0-9)	3	Les informations que le serveur affiche
MaxSession=entier	50	le nombre maximal de clients acceptés
Port=entier	210	le numéro de port utilisé
SAPI=chemin du fichier sapi.ini	Sapi.ini	le chemin d'accès au fichier sapi.ini
ServerType=	STANDALONE	type de serveur utilisé
TimeOut=nombre de secondes	3600	Déconnexion après un certain temps d'inactivité

Un exemple de fichier *zserver.ini* utilisé par le serveur Zserver est donné ci-dessous :

[DEFAULT]
AccessLog=/tmp/zserver_access.log
DRLIST=Traquair
DebugLevel=0
MaxSessions=50
Port=6668
SAPI=$HOME/Z39/sapi.ini
ServerType=STANDALONE
TimeOut=3600

Lancement du serveur

Le lancement du serveur se fait par la commande suivante :

Zserver [[-i Inifile][,groupe]]] [-oOption=value] [...]

Exemple: le serveur doit etre lancé sur le port 8888 avec un TimeOout de 500 secondes

Zserver –oPort=8888 –oTimeOout=500

B.2 L'API de recherche

Plusieurs applications nécessitent des fonctions de recherche de texte et de systèmes de bases de données. En réponse à ces besoins, CNIDR a développé une API de recherche (SAPI) qui a pour objectif de généraliser l'accès aux bases de données. Toute application qui utilise cette API pourra hériter des fonctionnalités des systèmes de bases de données au delà de l'API.

L'exploitation des bases de données se fait grâce à des fichiers textuels (sapi.ini) qui permettent de décrire la base de données en cours d'utilisation, son emplacement, les moteurs de recherche utilisés, le fichier faisant la correspondance entre les attributs Z39.50 standards et ceux utilisés dans la base de données.

Le fichier *sapi.ini* est constitué de groupes avec pour chacun une liste de directives. Sa structure est la suivante :

[GROUPE 1]
Directive 1=Valeur
Directive 2=Valeur
...
[GROUPE 2]
Directive 1 =Valeur
Directive 2 =Valeur
...

Dans ce fichier seront indiquées des informations concernant le type de moteur de recherche à utiliser (celui par défaut fournit par le système Isite ou un moteur de recherche personnalisé), le répertoire ou seront rangés les résultats d'une recherche, le fichier permettant de faire le lien entre la liste d'attributs standards et ceux définis pour la base de données. Ci-dessous est donné un exemple complet de fichier *sapi.ini* dans lequel est mentionnée l'utilisation du moteur de recherche par défaut pour une base de données et un script de recherche pour une autre.

[Default]	
DBList=ERIC,CATALOG	*les bases de données disponibles*
[CATALOG]	*la première base de données*
Location=/local/databases	*emplacement des indexes de la base de données*

[ERIC]	*la deuxième base de données*
Type=SCRIPT	*le moteur de recherche n'est pas standard*
Location=/usr/bin/Script_Search	*emplacement du moteur de recherche*
Results=/tmp/results	*où mettre les résultats*
Viewer=/usr/bin/Script_View	*script utilisé pour formater les résultats*
FieldMaps=mymap.ini	*fichier de correspondance entre attributs Standards Et ceux propre à la base de données*

Dans le fichier sapi.ini ci-dessus, nous avons deux bases de données, CATALOG et ERIC. La première est une base de données pour laquelle les outils et les paramètres par défaut sont utilisés, c'est à dire, le moteur de recherche proposé par le CNIDR. La seule information indiquée c'est l'emplacement des indexes de la base de données.

B.3 Configuration d'une base de données

Les deux types de moteurs de recherche supportés par SAPI sont ISEARCH et SCRIPT. ISEARCH est l'engin de recherche développé par le CNIDR. Il permet de faire des recherches booléennes et vectorielles. L'engin de recherche SCRIPT est le mécanisme qui permet d'intégrer d'autres engins de recherche qui accèdent à différentes bases de données.

Base de données SCRIPT

Le script permet d'implémenter des engins de recherche et d'accès à des bases de données particulières. A titre d'exemple, nous pouvons citer le cas d'une base de données SQL. Un script permettra de convertir les requêtes Z39.50 reçues en requêtes SQL afin d'interroger la base de données. Donc cette possibilité permet d'introduire de nouvelles bases de données sans étendre le code source de Isite.

La description d'une interface entre l'API de recherche et les applications externes a permis d'implémenter le processus d'intégration de nouvelles bases de données. Quatre éléments sont utilisés dans cette interface : le groupe d'informations sur la base de données, l'application externe, l'application qui formate les résultats et le fichier de résultats.

Le groupe d'informations : c'est une entrée dans le fichier sapi.ini qui décrit la base de données. Sur l'exemple suivant, on peut voir la description d'un groupe pour lequel ont été spécifiés le nom de la base de données, le type (SCRIPT), l'emplacement du script et le fichier des références des enregistrements trouvés et le script permettant de lire un enregistrement.

[ManPages]
Type=SCRIPT
Location=/usr/bin/ManPageSearch.sh
Results=/tmp/results
Viewer=/usr/bin/ManPageViewer.sh

Le fichier temporaire : lorsque l'API de recherche reçoit une requête dans la base de données *ManPages du* terme strcmp, la commande suivante est construite:

/usr/bin/ManPageSearch.sh /tmp/results.<pid> strcmp

Le fichier de résultats doit respecter une structure afin de permettre à l'API de lire les résultats.

Le format du fichier est le suivant :

Default
HitCount=3
Diagnostic=0
Separator=##separator string – your choice
[Data]
Record data for record number 1
##separator string – your choice

Record data for record number 2
##separator string – your choice
Record data for record number 3

Hitcount est le nombre d'enregistrements trouvés dans la base de données. La valeur 0 ou 1 dans la champ Diagnostic pour indiquer si la recherche a été fructueuse ou non, et la chaîne qui sépare les informations qui décrivent un enregistrement.

Base de données ISEARCH

ISEARCH est l'engin de recherche utilisé par défaut. Pour l'utiliser il faut indiquer dans le groupe décrivant la base de données, le répertoire où se trouvent les fichiers d'indexation de la base générés avec l'outil d'indexation *Iindex* fournit avec *Isite*

Exemple :

[Default]
DBLIST=CATALOG

[CATALOG]
Location=/local/databases

Une base de données de type ISEARCH peut utiliser le champ FieldMaps afin de faire des recherches conformes au standard Z39.50. Un fichier est utilisé pour indiquer la correspondance entre les attributs Z39.50 et les informations de la base.

Exemple :

Supposons que nous avons dans la base de données des fichiers de type HTML et qu'ils ont été indexés selon les TAG standards du langage et que nous utilisions l'ensemble des attributs bib1.

Un fichier de la base de données :

<TITLE> Document numéro 1</TITLE>
<H1> le sujet du document </H1>
<BODY> le texte... </BODY>

le fichier de correspondance Mymap.ini est le suivant :

[Default]
bib1/21=H1

Une recherche standard selon l'attribut 21 sera interprété comme étant une recherche selon l'attribut H1 pour la base de données courante.

Le fichier sapi.ini sera le suivant :

[Default]
DBLIST=CATALOG

[CATALOG]
Location=/local/databases
FieldMaps=mymap.ini

Lorsque le serveur est lancé, il lira le fichier mymap.ini est surchargera la valeur de bib1/21 et fera la correspondance avec le champ de nom H1.

B.4 Le système de recherche Isearch

L'engin de recherche Isearch fournit avec le système Isite se base sur un outil Iindex pour l'indexation de différents types de fichiers qui constitueront la base de données. Nous commençons la description de cet outil puis expliquerons le fonctionnement de Isearch.

Indexation d'une collection

L'outil d'indexation prend en entrée une liste de fichiers à indexer et génère des fichiers d'indexes. La base de données est constituée des fichiers source et des fichiers indexes.

Il est possible d'indexer différents types de fichiers. Doctype est l'option utilisée dans la commande d'indexation. Cette option permettra de savoir comment rechercher les documents logiques et les champs dans les documents. Par défaut le type SIMPLE est utilisé. Ce type indique à Iindex qu'il existe un document logique par fichier et qu'il n'y

a aucun champ d'indexation. Dans ce cas, les fichiers sont considérés comme de simples fichiers texte.

SGMLTAG est le deuxième type de fichier que Iindex peut indexer. Les indexes sont générés en fonction des champs (TAG) SGML. A un fichier est associé un document logique. HTMLTAG est le type utilisé pour les fichiers HTML pour indexer les pages HTML pour la recherche de sites WWW. D'autres types de documents existent et le CNIDR en crée constamment de nouveaux.

Les options de la commande Iindex sont les suivantes :

-d (X) : spécifier le nom du répertoire des indexes.
-a : ajouter de nouveaux fichiers index à ceux qui existent déjà.
-m (X) : le nombre de Mega octets utilisé pour l'indexation.
-s (X) : indique qu'il existe plusieurs documents logiques par fichier physique.
-t (X) : indique le doctype à utiliser.
-f (X) : indique la liste de fichiers à indexer.
-r : indique qu'il faut descendre récursivement dans les sous-répertoires.

Exemple :

Iindex –d NomBase –t HTMLTAG –f /local/fichier/ .**

Permet d'indexer tous les fichiers qui se trouvent dans le répertoire /local/fichiers avec le doctype HTML. Le résultat de l'indexation sera mis dans le répertoire NomBase.

La recherche d'information

1 - recherche simple:

recherche les documents qui contiennent une liste de mots.

Isearch –d NomBase petite maison

2 - Recherche dans des champs:

recherche de documents qui contiennent des mots dans certains champs

Isearch –d Nombase title/maison

Permet de rechercher les documents qui contiennent le mot maison dans le champ titre.

3 - Recherche booléenne:

Deux types de notation existent. Il y a le Infix et le RPN. Nous en donnons quelques exemples.

Format Infix : Isearch –d Nombase –infix maison Or prairie
Format RPN : Isearch –d Nombase –rpn maison prairie Or

4 - Utilisation des caractères joker

Le caractère * peut être utilisé pour faire des recherches avec troncature. La requête:

*Isearch –d Nombase cha**

permettra de rechercher les documents qui contiennent les mots qui commencent par ``*cha*``

5 - Classification des documents retrouvés

L'algorithme de Gerald Salton est utilisé pour faire la classification des documents retrouvés suite à une recherche. Les documents qui contiennent le plus d'instances et dont les mots ont un score élevé sont rangés en début de liste.

Il est possible de faire des recherches en affectant des poids aux mots utilisés dans la requête. Les mots ont le même poids par défaut.

Exemple :

Isearch –d Nombase maison :5 prairie

Cette requête indique à Isearch des recherches les documents qui contiennent les mots maison et prairie. Le poids donné au mot maison est cinq fois plus élevé que dans le cas d'une recherche normale.

La requête suivante fait le contraire :

Isearch –d Nombase maison : -5 prairie

Rechercher les documents qui contiennent maison et prairie. Soustraire 5 du score des documents qui contiennent maison. Donc les documents qui contiennent le mot prairie sont plus important puis viendront les documents qui contiennent prairie et maison, enfin les documents contenant le mot maison.

B.5 La passerelle http / Z39.50

Isite offre la possibilité d'établir une session entre un serveur http et un serveur Z39.50. Sur la figure 1, la combinaison d'un serveur http, zgate et zcon, représente la passerelle qui s'exécute sur une machine unique. Un fureteur WWW se connecte au serveur http et utilise une forme HTML pour transmettre les informations concernant une session Z39.50. Le CGI zgate analyse les informations envoyées et selon le cas, lance une connexion zcon s'il n'en existe pas une déjà en exécution pour la session en cours. Par la suite, toutes les requêtes sont passées de zgate à zcon qui lui entre en communication avec le serveur Z39.50. Les résultats sont transmis au CGI zcon puis à zgate, enfin au serveur http pour être envoyés à l'usager sous la forme de page HTML. Le processus zgate se termine alors à ce moment là alors que le processus zcon reste en vie durant toute la durée de la session. [CNI]

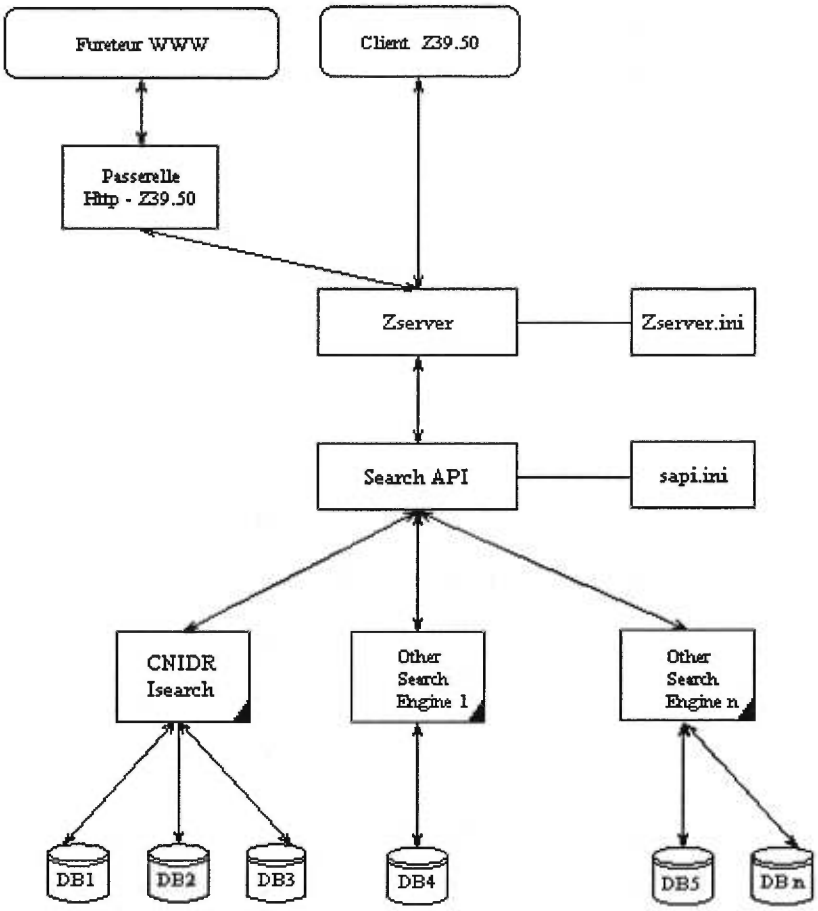


Figure B1 : Arcitecture de la psserelle http / Z39.50