Stage EDF R&D - CNRS/Limsi

DESS Ingénierie Informatique - Université Paris XI - Orsay.

RABIAZA Nirina Anthony

Stage EDF R&D - CNRS/Limsi:
DESS Ingénierie Informatique - Université Paris XI - Orsay.
RABIAZA Nirina Anthony

Remerciements

Je tiens à remercier :

- Toute l'équipe du groupe *SOAD - EDF R&D*
- Toute l'équipe du *Limsi*
- Benoît Habert, chercheur au *Limsi*, pour son encadrement, sa confiance et son enthousiasme (et également ses tablettes de chocolats)
- Helka Folch, post-doctorante au *Limsi*, pour son encadrement, sa disponibilité et ses blagues
- Sylvaine Nugier, responsable dans le département *SOAD - EDF R&D*, pour son encadrement et son enthousiasme
- Yasmina Quatrain, du département *SOAD - EDF R&D*, pour sa collaboration
- Jean Vidal, responsable de la *branche Commerce - EDF*, pour sa collaboration
- Mathieu Brugidou et Dominique Leroux, du *GRETS - EDF R&D*, pour leur collaboration
- David Leray, du *projet p000f* et en DESS SCHM *Paris XI*, pour sa collaboration
- Nicolas Renet, du *Limsi-projet p000f*, pour sa collaboration
- les responsables de l'Université *Paris XI - Orsay*, qui m'ont permis la réalisation de ce stage
- Solo-Lalao et Voahangisoa pour leur support, malgré les 10,000 km qui nous séparent
- Christel, Valérie, Carly et Alex pour leur aide précieuse
- Tous ceux qui ont contribué à l'accomplissement du projet p000f et à la complétion de mon stage

Table des matières

Liste des illustrations

Introduction

Mon stage a été effectué dans le cadre de mon DESS *Ingénierie Informatique*, formation suivie à l'Université Paris XI - Orsay. Ce stage de 6 mois a été effectué au sein de la direction *Recherche et Developpement* du groupe **EDF** et dans les laboratoires du **Limsi**.

Une des nombreuses missions du pôle Recherche et Developpement d'EDF est de solutionner les problèmes rencontrés par les différentes branches EDF. Ces branches utilisent leurs propres entrepôts de données (des bases relationnelles, des bases objets, des tables Access, ...). Ces entrepôts capitalisent les "connaissances" du groupe, nous pouvons y trouver des bases de données clientèles, des données issues du web, des bases d'enquêtes ou encore des bases d'entretiens. Le département SOAD (Statistique et Outils d'Aide à la Décision) effectue des études sur les bases clientèles afin de connaître au mieux chaque client EDF, ceci afin de lui proposer des services et des produits adaptés à ses besoins. Les logiciels de *datamining* et les différents outils d'analyse statistiques prennent en entrée des sources homogènes et l'hétérogénéité des différentes bases EDF interdit toute analyse : nous devons *Structurer ces données complexes* selon des standards *XML* pour pouvoir permettre la réalisation du *datamining*.

Le premier chapitre de ce rapport nous présente le Groupe *EDF* et la coopération du département *SOAD* avec le *Limsi* pour la réalisation du *projet p000f*. Le second chapitre nous énonce les étapes de cette structuration et les raisons qui nous ont motivées pour le faire. L'analyse des besoins et le cahier des charges sont exposés dans le chapitre 3. Dans le quatrième, nous verrons en détail les types de données rencontrées. Dans la cinquième partie, nous allons découvrir les différents *langages de description* que nous avons choisis pour structurer les entrepôts et les raisons de ces choix. Viendront ensuite (Chapitre 6) les outils utilisés pour l'implémentation de l'application et l'application en elle-même : ses modules et son interface graphique (Chapitre 7), nous conclurons par les perspectives d'évolutions (Chapitre 8) de l'application.

Chapitre 1. Contexte du stage

1. Stratégie et structure du Groupe EDF

Le Groupe EDF est un énergéticien intégré [1] constitué d'un réseau d'entreprises. Leader dans la production et la distribution d'électricité, il gère une puissance installée de près de 122,6 GWh. Le Groupe EDF fournit énergie et services à 41,6 millions de clients dans le monde dont 35,6 millions en Europe. En 2003, il a réalisé un chiffre d'affaire de 44 919 millions d'euros. Le Groupe est ainsi un des leaders européens de l'énergie. Sa croissance maîtrisée s'oriente vers de nouveaux marchés et de nouvelles activités. Le principal marché d'EDF reste le territoire français mais le Groupe étend sa stratégie dans un contexte de concurrence internationale. L'ouverture du marché de l'énergie à la concurrence est, pour EDF, un défi et l'opportunité de devenir un énergéticien de référence en Europe. En France, le marché énergétique a été ouvert dés 1999, avec une première tranche de clients éligibles [2] (consommation électrique minimale de 100 GWh par an) élargie en janvier 2000 à 30% du marché (consommation minimale de 16 GWh par an). Depuis le 1er juillet 2004, le marché de l'électricité est ouvert à tous les clients professionnels ; et le sera le 1er juillet 2007 pour les particuliers. La France se prépare ainsi à l'ouverture à 100% de son marché de l'électricité à la concurrence. Comme la plupart des grands groupes européens de ce secteur, EDF est en évolution. Cette évolution a été initiée depuis plusieurs années par la mise en place d'une nouvelle organisation (organigramme ci-après) qui devrait favoriser également le développement de nouveaux métiers et de nouveaux services permettant ainsi au Groupe de demeurer l'un des leaders du marché européen.

Les principales entités France sont les suivantes :

- La Branche **Production - Ingénierie** regroupe les compétences et les actifs nécessaires à l'exercice du métier de manager d'énergies ;
- La Branche **Management d'Energies** est chargée d'assurer la gestion stratégique de l'équilibre des portefeuilles amont-aval et son optimisation ;
- La Branche **Commerce** rassemble les métiers de vente d'énergie aux entreprises, aux collectivités locales, aux professionnels et aux résidentiels ;
- La Branche **Collectivités - Distribution** assure le fonctionnement du réseau de distribution français d'électricité (lignes à Basse Tension et Moyenne Tension), ainsi que la qualité des services associés, et ceci pour l'ensemble des utilisateurs du réseau.
- Le **RTE** assure la gestion du réseau de transport de l'électricité (lignes à Haute Tension et Très Haute Tension).

[1] Etre un acteur intégré, c'est être présent en amont et en aval de la chaîne énergétique : de la production à la commercialisation. Le transport et la distribution sont quant à eux deux activités régulées qui peuvent être ou non exercées par les acteurs présents en amont et en aval.

[2] Un client est dit éligible dès lors que sa consommation électrique annuelle est supérieure à un seuil fixé en GWh

Concernant le statut de l'entreprise (ainsi que celui de Gaz de France), le Ministère de l'Economie, des Finances et de l'Industrie a travaillé à sa transformation. Jusqu'alors EPIC [3] , la loi n°2004-803 du 9 août 2004 relative au service public de l'électricité et du gaz et aux entreprises électriques et gazières (publiée le 11 août au Journal officiel) prévoit le changement de forme juridique des deux entreprises **EDF** et **Gaz de France** en sociétés anonymes. C'est un décret d'application, devant être passé avant le 31 décembre 2004, qui fixera les statuts initiaux de la « **Société EDF** ». La transformation en **société anonyme** sera alors seulement effective.

2. La direction Recherche et Développement

La direction EDF Recherche et Développement (EDF R&D) dépend de la Direction Générale Opérations. Elle est répartie sur trois sites principaux situés en Ile-de-France : Chatou (78), Les Renardières (77) et Clamart (92). Forte de 2650 chercheurs et techniciens, la R&D dispose de laboratoires du plus haut niveau international en matière de génie électrique, d'études de matériaux, de mécanique, de thermique ou d'hydraulique, appuyés par d'importants moyens de simulation numériques. Le travail de la R&D s'articule autour de plusieurs domaines d'activité : le développement commercial, la production d'électricité (nucléaire, thermique à flammes, hydraulique, énergies renouvelables), les réseaux électriques (infrastructures et développement des réseaux de transport, fonctionnement et conduite du système, réseaux et ouvrages de distribution). La R&D travaille également sur des domaines transverses tels que les technologies de l'information et de l'environnement. Le site de Clamart regroupe la moitié des forces de la recherche d'EDF. Il réunit surtout des activités à caractère non expérimental et des moyens de calcul relatifs à l'informatique scientifique.

3. Le département ICAME

EDF R&D s'organise autour de seize départements centrés sur des axes de recherches différents. Parmi eux, le département ICAME (Innovation Commerciale, Analyse des Marchés et de leur Environnement), créé début 2002, met en œuvre des compétences diverses (sociologie, marketing, finance, informatique, data mining et statistique), enrichies par une connaissance et une compréhension :

- de l'environnement du Groupe EDF (politique, social, économique, réglementaire et institutionnel) ;
- des différents segments de marchés (volume, marges, concurrence, axes stratégiques de développement du Groupe) ;
- des clients (perceptions, comportements, consommations, satisfaction, attentes en matière de services) ;
- de processus de relation clientèle.

3Etablissement public à caractère industriel et commercial

Le but de ceci étant de développer des offres tarifaires et les services associés, en appui de la Branche Commerce. Ce département est composé de sept groupes, certains à dominantes « métier » dédiés à un marché particulier (par exemple le marché résidentiel), d'autres à dominantes « compétences » chargés de développer de nouvelles méthodes.

4. Le groupe SOAD

L'ambition du groupe SOAD (Statistique et Outils d'Aide à la Décision) est d'être une force d'innovation, de proposition et d'action dans le domaine des Systèmes d'Information d'Aide à la Décision ; il doit favoriser l'exploitation des banques de données de l'entreprise par les outils mathématiques et informatiques, à des fins d'une meilleure prise de décision vis à vis de la clientèle, du réseau ou du parc de production. Les domaines techniques particulièrement explorés sont la statistique, les mathématiques pour l'aide à la décision, les outils d'analyse, d'extraction et de présentation de l'information, les outils d'accès en ligne à l'information dans les entrepôts de données. Les domaines de compétence du groupe SOAD s'articulent autour de deux axes principaux :

- Les méthodes d'analyse statistique (analyse exploratoire et décisionnelle de données, techniques de data mining pour l'analyse des courbes de charge, etc.).
- L'accès analytique aux entrepôts de données (vérification de la cohérence des données, explication des incohérences, construction de cubes de données, OLAP, etc.)

Pour chacun de ces deux axes, le groupe a pour ambition de mettre en œuvre une double logique d'innovation (exploration, mise au point, évaluation de nouvelles méthodes, études de nouvelles données, intégration de différentes techniques, etc.) et de transmission de savoir-faire (formations, animations statistiques, transfert de compétences, conseil et expertise, etc.) au service de l'ensemble de l'entreprise.

5. Le CNRS-Limsi

Le LIMSI est une Unité Propre de Recherche du CNRS, associée aux universités Pierre et Marie Curie (Paris-6) et Paris-Sud (Paris-11). Le laboratoire accueille environ 120 permanents (chercheurs, enseignants-chercheurs et ITA-IATOS) et une soixantaine de doctorants. Il mène des recherches pluridisciplinaires en Mécanique et Energétique et en Sciences et Technologies de l'Information et de la Communication, couvrant un large spectre disciplinaire allant du "thermodynamique au cognitif", en passant par la mécanique des fluides, l'énergétique, l'acoustique et la synthèse vocale, le traitement de la langue parlée et du texte, la vision, la réalité virtuelle... Le LIMSI a l'ambition et la volonté d'allier excellence disciplinaire dans l'ensemble de ses champs scientifiques et développement de projets pluridisciplinaires, en partenariat avec d'autres organismes de recherche et des industriels, français et étrangers. Il s'investit tout particulièrement dans la formation par la recherche et la diffusion de l'information scientifique et technique.

Mon stage s'est effectué en collaboration avec le département Communication Homme-Machine du **Limsi** (groupe Langues, Information et Représentations). *Benoît Habert* et *Helka Folch* ont en été les pivots de cette collaboration **EDF R&D - Limsi**.

6. Présentation du Projet "p000f"

Au sein du groupe SOAD, le projet « méthodes et outils d'analyse de données clientèle » (p000f) fédère tous les travaux ayant pour objectif d'explorer ou d'éprouver les technologies émergentes permettant de valoriser les données clientèle en vue d'améliorer l'efficacité des directions opérationnelles d'EDF. Plus précisément, le projet p000f s'organise autour de trois thèmes d'études :

- Préparation, manipulation et représentation formelle des données ;
- Analyse et fouille de données structurées et complexes ;
- Restitutions des données et des résultats d'analyse ;

7. Problématique

Le groupe EDF est composé de différents départements, chacun de ces départements capitalise ses données clientèles dans des entrepôts de données spécifiques. Ces différents entrepôts de données capitalisent des connaissances provenant d'univers différents, celles-ci peuvent être associées et de cet agrégat pourront être tirées de nouvelles connaissances.

Malheureusement, fusionner ces entrepôts de données n'est pas aisé. Ils présentent des structures différentes et aucune application, à l'heure actuelle, n'est capable de les explorer et faire leurs rapprochements. De ce fait, il est nécessaire de "traduire" ces entrepôts dans une "langue" commune.

Le stage **Structuration XML de données complexes pour le datamining** s'insère dans les travaux liés au thème **Préparation, manipulation et représentation formelle des données** du projet p000f.

Nous allons voir, dans le chapitre suivant, les motivations de la structuration de ces données et les différentes étapes que nous allons suivre pour effectuer cette structuration.

Chapitre 2. Structuration XML des données complexes pour le text-mining

1. Les étapes de la formalisation des Entrepôts de données

Les différents responsables *EDF* et *CNRS-Limsi*, après plusieurs réunions, ont convenus les étapes suivantes pour la réalisation du stage :

1. Connaître et répertorier les différents types de documents utiles à la fouille de données à EDF : entretiens, forums, données issues du Web, réunions, notes de recherche, bases de données clientèles, enquêtes. Établir la source de ces données (base de données DEGS, enquêtes direction de marketing, etc.) et par qui elles sont utilisées (département *ICAME*, etc.).

2. Sélectionner les sources intéressantes en examinant les informations et méta-informations qu'elles contiennent.

3. Modéliser ces différentes sources sous forme XML ; des diagrammes faciliteront la lecture des différents modèles. Établir une DTD ou un schéma pour valider les données textuelles transformées.

4. Assurer l'articulation entre les données textuelles et les métadonnées; Le travail porte ici sur les métadonnées primaires au sens de variables signalétiques ou des champs de bases de données en opposition au métadonnées secondaires au sens de résultats d'analyses. (Ex : Assurer le lien entre le contenu des enquêtse/entretiens et les catégories de codages utilisés par les sociologues).

5. Implémenter l'interface qui prenne en charge les différents formats hétérogènes d'entrée (Bases Lotus Notes/Domino, bases Access, tables SAS, fichiers textes, etc.) et les normaliser selon un modèle XML (DTD ou schéma).

6. Réaliser un système de validation des fichiers XML qui permet de repérer les erreurs et dans le cas échéant de corriger celles qui ont été décellées.

1.1. Les données

La direction Marketing, le département *ICAME* et le *GRETS* ont spécialement constitué une collection de documents pour l'élaboration du prototype.

Voici quelques caractéristiques de cette collection :

- Documents très variés structurellement, avec des liens portant sur les informations contenues dans les bases de données clientèles.
- Documents ayant des thèmes communs, ceux-ci sont importants pour la chaîne d'analyse et pour le repérage des points de vue dans un corpus.

1.1.1. Données Structurées

Voici une liste des différents types de données structurées *EDF* qui ont paru adaptés dans la mise au point d'un prototype:

- La direction *Marketing* a mis à notre disposition la base Dixit : il s'agit d'un ensemble d'enquêtes satisfactions, d'entretiens et de tables rondes. Ces données sont classées par marché (particuliers, professionnels, PME-PMI, industriels, etc.) et sont contenues dans une base Lotus Notes.
- Les bases de données clientèle Optimia de la *DEGS* : ces bases contiennent pour les clients particuliers et professionnels d'*EDF* des variables (consommation, département, code NAF, type d'habitation, etc.) et des champs commentaires (saisis par les opérateurs EDF durant des conversations téléphoniques avec le client). Ces bases de type Oracle ne sont pas directement interrogeables mais le département *ICAME* possède une extraction de la base qui correspond aux clients d'un (1) centre *EDF* (approximativement 400 000 clients) sous forme de tables SAS.

Le département *ICAME* nous a également délivré différentes sources de données :

- Des données issues d'un forum électronique de discussion qui permettront la réalisation d'un premier prototype et des premiers tests de "requêtage".
- Des questions ouvertes et des baromètres satisfaction : il s'agit d'un ensemble de questions fermées (QCM) et ouvertes posées à un large échantillon d'individus (environ 1500 personnes par enquête). Les réponses à ces QO/BSM[4] sont stockées dans des tables SAS.
- Des entretiens téléphoniques effectués par des sociétés spécialisées dans ce domaine : ce sont des données ayant une structure similaire à celle des QO/ BSM. Ces entretiens sont sous forme de fichiers Access.
- Les questionnaires relatifs aux entretiens téléphoniques. Ils sont sous forme

[4] QO/BSM : Questions ouvertes/Baromètres Satisfaction Marketing

textuelle mais respectent tout de même une structure donnée (cf Enquêtes Téléphoniques). Ces questionnaires sont au format MS Word.

1.1.2. Données Textuelles

Outre les données structurées, les divers entrepôts *EDF* contiennent également des données non structurées : de type textuel ne possédant pas de véritable structure. Voici les différents types de données textuelles rencontrées :

- Des guides d'entretiens que l'on peut trouver dans la base Dixit. Ces guides correspondent aux différentes étapes à suivre lors d'un entretien, ils ont été écrits lors de la définition de l'enquête. Ils existent sous divers formats : MS Word ou MS PowerPoint.

- Les entretiens individuels (base Dixit); Il s'agit en fait du dialogue (les tours de paroles) entre l'interviewer et la ou les personnes interviewées : Ce sont les questions posées et les différentes réponses données ; il arrive parfois de trouver des commentaires concernant des réactions de l'interviewé ou des actions (Ex : "Rire" ou "Sa femme rentre..."). Ces entretiens sont en format MS Word

- Les entretiens de groupe (ou table ronde) également contenus dans la base Dixit: de la même manière que dans les entretiens individuels, nous avons les tours de parole des différents intervenants. On trouve ces entretiens en format MS Word.

- Un large corpus sur les controverses "lignes de haute tension" venant du *GRETS*. Celui-ci est constitué d'environ 40 heures d'entretiens (soit 40 000 mots) dont on possède l'enregistrement et la retranscription.

2. Pourquoi structurer ces données complexes?

Dans cette partie, nous vous proposons de spécifier les raisons motivant cette "structuration des données complexes".

2.1. Analyse combinée : données textuelles et méta-données

Une fois que les entrepôts de données seront formalisés, nous pourrons effectuer diverses analyses sur les ces entrepôts. Une recherche de type "Full text" sera possible sur tous les documents; encore plus intéressant, du fait que nous disposons de méta-données sur ces données, nous pourrons :

- faire des recherches sur les méta-données

- effectuer une recherche sur les documents après les avoir filtrer à l'aide des méta-données (analyse combinée)

Ainsi, il sera possible de caractériser le resultat des analyses textuelles (Ex: Analyses des questions ouvertes d'une enquête) avec des méta-données (Ex: Information sur la personne enquêtée : age, consommation électrique,...). Ce type d'analyse peut, par exemple, avoir comme finalité la construction d'une typologie des clients ou celle de leurs comportements.

2.2. Analyse statistique

La fusion statistique est la combinaison de données, provenant de sources différentes, pour obtenir un seul jeu de données dans lequel toutes les variables sont renseignées (présence obligatoire de variables communes).

Une fois les données complexes structurées nous pourrons résoudre un problème majeur qui se pose chez EDF : l'ensemble des informations sur chaque client n'est pas disponible dans une unique base de données, il y a des données manquantes (variables internes) et le plusieurs variables (externes) sont absentes des bases de données clientèle EDF.

Cette formalisation permettra alors à EDF de mieux connaître le client afin de lui proposer des services et des produits adaptés à ses besoins. L'analyse statistique de sources différentes permettra de faire de l'*analyse stratégique* : nous pourrons alors avoir des points de vues différents sur le client et nous pourrions alors contruire une vue globale de son comportement.

2.3. Capitalisation et Analyse secondaire

Cette formalisation des entrepôts nous permettra une *capitalisation des connaissances*. Ces connaissances structurées seront stockées sous forme de fichiers homogènes **accessibles** et **réutilisables**. Pour ce fait, ils doivent posséder une structure respectant un ou plusieurs standards, cela faciliterait l'automatisation et la généralisation des procédures de "requêtage" et d'analyse des données.

La capitalisation des connaissances requiert la formalisation des données structurées classiques (base de données relationnelles, ...), mais également de données plus riches, en particulier les données textuelles (entretiens, forums, ...); un autre point important est celui de la création des *liens* entre ces sources de données hétérogènes.

Les résultats d'une étude **x** seront alors capitalisés et il sera possible de réexploiter ces données pour répondre à d'autres études. Un exemple a été vu lors du stage: des analyses secondaires ont pu être effectuées sur d'anciennes études pour répondre à une question jamais posée : "Quelles sont les utilisations détournées du chauffage électrique?". A partir de cet exemple, nous voyons l'importance des analyses secondaires et l'économie en temps et en argent que cela peut apporter.

Chapitre 3. Analyse des besoins

L'analyse des besoins a été une des missions importantes que l'on m'a confiée durant le stage. Effectivement, c'est sur celle-ci que repose l'intégralité du projet et premièrement la rédaction du cahier des charges.

Afin de connaître les désirs de chaque département, notre équipe a décidé d'établir un questionnaire précis. Celui-ci nous permettra de cerner les activités du département et recueillir ses besoins.

1. Questionnaire

Voici un résumé du questionnaire que l'on a utilisé durant l'entretien avec les principaux commanditaires et futurs utilisateurs de la Plateforme que notre équipe devait concevoir :

1. Profil de l'utilisateur

    - Ses compétences informatiques, les logiciels qu'il utilise,...
    - Ses besoins ou attentes par rapport à ce projet,
    - Ses habitudes de collaborations.

2. Traitement des données

    - Le type de données, la manière dont on y accède?
    - L'existence de modèles, nomenclatures, etc.
    - Utilise-t-il des fonctionnalités de réutilisation de données entre logiciels?

3. Traitement des résultats

    - Qu'appelle-t-on « résultat » ?
    - Quels sont les formats utilisés?
    - Y a-t-il dialogue avec les autres services?

Cet entretien sera semi-dirigé et adapté au métier des interlocuteurs (sociologie, datamining, marketing).

2. Entretien avec la Branche Commerce

Un premier entretien s'est tenu avec un responsable de la branche *Commerce* : *Jean Vidal*. Celui-ci dirige le département *Régulation et Étude*s. Il y joue le rôle de consultant interne pour plusieurs services, dont celui du *Marketing Stratégique* : au moment de passer commande d'une enquête, ce service fait appel à son expertise.

A la suite de cet entretien, voici des informations utiles qui serviront à l'établissement du cahier des charges.

Différentes étapes de la réalisation d'une étude

Un client interne désirant l'élaboration d'une nouvelle étude s'adresse au chargé d'affaire du département Régulation et Études. Ce département va recueillir l'expression des besoins, construire le cahier des charges en vérifiant l'existence éventuelle d'études similaires, préparer la démarche d'achat, accompagner le commanditaire tout au long de l'étude et finalement capitaliser les résultats.

L'équipe de Jean Vidal propose les types d'études les plus adaptés au besoin du client interne. Ils élaboreront ensemble le cahier des charges qui comporte généralement les points suivants :

- Contexte de l'étude
- Objet
- Périmètre
- Propositions de méthode

(Ex : Étude sur les clients professionnels, 10 entretiens qualitatifs et 3 tables rondes)

Au moment de la définition du cahier des charges, les sociologues et marqueteurs peuvent consulter une base MKM (Marketing Knowledge Management) qui capitalise les "connaissances marketing" (documents internes/externes, études précédentes, etc.) ou consulter des BSM (Baromètres Satisfaction Marketing) ; ces secondes couvrent l'ensemble de la relation marché/client. Ils sont constitués d'un grand nombre d'entretiens répondant à des questions ouvertes et fermées.

Ces différentes bases sont interconnectées et des liens permettent de naviguer dans celles-ci (Ex : des pointeurs permettent l'accès aux données brutes d'une étude depuis son rapport final).

Une fois le cahier des charges rédigé, un appel d'offres déterminera l'institut de sondage le plus adéquat. Ce dernier réalisera ensuite le nombre souhaité d'entretiens et effectuera la transcription; des référentiels et vocabulaires lui sont donnés par EDF pour la définition des populations et celle des variables d'enquête ; nous avons par exemple les nomenclatures INSEE (Ex : code NAF5 pour le codage

des activités) ou encore des nomenclatures propres à EDF (Ex: les segments de marché).

Le sous-traitant remet à la fin de l'étude une présentation PowerPoint ou un document Word contenant les résultats de l'étude et des données quantitatives au format SAS.

Organisation des données

Les données sont stockées dans différentes bases :

- la base Dixit (cf Entretiens Capitalisés) : une base Lotus Notes contenant une base d'entretiens qualitatifs ; Pour chaque entretien, nous possédons une fiche signalétique structurée et un document Word attaché en pièce jointe qui correspond à l'entretien à proprement.
- des fichiers de données SAS avec leurs codebook (lien entre le questionnaire et les données) : il s'agit du résultat des enquêtes sous forme de tableaux avec les champs des colonnes correspondant aux questions (cf Export SAS).

Compétences Informatiques des utilisateurs

Au sein de la branche commerce, on peut distinguer deux profils d'utilisateurs :

1. L'équipe *Régulation et Études*

   Les chargés d'affaires de ce département ont un rôle central dans la capitalisation des résultats : ils réalisent des travaux intermédiaires comme le retour dans les données quantitatives, les extractions ou les croisements de base. Leur rôle est principalement de répondre à des questions du département *Marketing* (Ex : le discours des locataires et celui des propriétaires est-il le même?). Pour cela, Ils utilisent notamment Alceste (logiciel de text-mining) ou SPAD (logiciel d'analyse de données et data-mining).

   ## Note

   Alceste est un logiciel d'analyse de données textuelles qui réalise sur un corpus une classification descendante hiérarchique d'unités textuelles élémentaires. Les classes (ou thèmes) dégagées par le logiciel sont décrites par des mots caractéristiques, par des phrases typiques, par une représentation spatiale (analyse factorielle) de ces classes et par différents indices associés.

2. L'équipe *Marketing*

   Les marqueteurs commandent des études, relisent les entretiens et effectuent des extractions de verbatims. Pour la réalisation de ces tâches, tous utilisent la

[5] NAF : Nomenclature d'Activités Française

suite Office (MS Word, Excel et PowerPoint). Certains possèdent des connaissances Access ou maîtrisent un logiciel de traitement statistique (SPSS[6]). Il leur arrive également d'utiliser des croiseurs (Ex : tris dynamiques croisés) et pour cela ils utilisent *SAS*.

Tous travaillent sous Windows et utilisent principalement des outils bureautiques : il sera donc nécessaire que la nouvelle application ait une interface simple, intuitive et que celle-ci fonctionne sous Windows. Cette application devra assurer la conversion de la base Lotus Notes : structurer données (guide d'entretiens et entretiens) et métadonnées (informations sur l'interviewé) tout en assurant le lien entre ces dernières

3. Entretien avec les sociologues du GRETS

Un entretien a été effectué avec deux sociologues du département GRETS :

Mathieu Brugidou et *Dominique Leroux*. Ils organisent et réalisent les études commanditées par divers départements. Chaque sociologue possède ses propres méthodes de travail et capitalise le résultat de ses enquêtes différemment.

Capitalisation des études

Les sociologues gardent la retranscription des entretiens en format électronique et en format papier, ils disposent également d'une grille d'analyse pour le codage des données. Ils capitalisent les entretiens dans un verbatim : une base Notes d'une structure similaire à celle de la base Dixit.

Analyses primaires et secondaires

Les analyses primaires sont généralement effectuées avec des systèmes d'assistance au dépouillement de données qualitatives ou CAQDAS[7]. La mise en oeuvre de ce type d'outils nécessite un archivage structuré des entretiens.

Les analyses secondaires sur les données ne sont pas couramment effectuées. En effet, il ne s'agit pas d'une pratique courante et son utilisation est assez limitée : on peut facilement sortir du contexte et les résultats obtenus dépendent fortement de la manière dont les entretiens ont été retranscrits.

4. Entretien avec le département Data-Mining et Statistiques

Plusieurs réunions ont été faites avec *Sylvaine Nugier* (responsable du SOAD) et *Yasmina Quatrain*. Elles nous ont présenté les données que le département utilise

6SPSS : Statistical Package for the Social Sciences
7CAQDAS : Computer Aided Qualitative Data Analysis Systems

pour effectuer leurs analyses et leurs méthodes de travail.

Type des données

Le SOAD dispose d'un très grand nombre de source de données : données issues du Web, base Access, tables Excel, tables SAS, etc. L'hétérogénéité de ces sources complique considérablement l'analyse de celles-ci ; En effet, les outils d'analyses statistiques prennent en charge des formats particuliers et il était alors impossible, avant la réalisation de ce projet, d'analyser convenablement certaines données.

Deux sources de données posaient effectivement des problèmes :

- Des données issues du Web : un forum de discussion extrait d'une base *Lotus Domino*. Ces données extraites de la base de données Objets d'IBM, malgré le fait qu'elles soient structurées, ne permettent pas la mise en oeuvre d'analyses statistiques.

- Les questionnaires CATI composés d'un questionnaire produit par un logiciel d'Interviews téléphoniques et de données *Access* correspondant aux réponses des interviewés. La séparation du questionnaire et des données pose problème : il est difficile depuis ce second de déterminer a quoi correspond la colonne **x** ou la réponse **y** car ces informations sont uniquement décrites dans le questionnaire.

Outils utilisés

Le département SOAD utilise principalement des outils d'analyse statistique : SAS et des logiciels de data-mining comme Alceste, Temis Insight Discoverer[8].

Tâches à effectuer

Les outils que nous allons mettre en oeuvre dans ce projet devront permettre :

- De structurer les données issues du Web et des bases de capitalisation d'entretiens et permettre la visualisation, le contrôle et la modification des documents structurés.

- De structurer les questionnaires CATI et ainsi assurer le lien entre questionnaire et données. Une fois les données structurées, il sera alors possible d'effectuer une exportation au format SAS : format permettant l'analyse de celles-ci.

8www.temis-group.com

5. Élaboration du cahier des charges

A partir des enquêtes et analyses réalisées préalablement, nous avons pu instaurer en concertation avec *Helka Folch* (CNRS-Limsi) un cahier des charges. Celui-ci relate les démarches à suivre dans le but de réaliser le projet de façon optimale.

Le prototype qu'il a été demandé de concevoir devait assurer les 4 fonctions suivantes :

- La **transformation** des différentes sources en documents RDF/XML.

  3 sources ont été choisies parmi les entrepôts EDF :

  1. Un forum de discussion (extrait d'une base de données Lotus Domino)
  2. Les enquêtes téléphoniques (base Access)
  3. Une base de données d'entretiens capitalisés (base Lotus Notes)

- La **visualisation** des documents RDF/XML générés.

  On peut naviguer dans les documents RDF/XML d'un forum de discussion, d'une enquête ou de la base données d'entretiens capitalisés à l'aide de différentes interfaces; Les interfaces sont différentes et adaptées au type de document (affichage sous forme de tableau, de menus arborescents, affichage de la même manière qu'un traitement de texte, ...)

- La **validation** de ces documents et l'affichage des erreurs.

  Les documents RDF sont contrôlés et les erreurs affichées dans l'interface.

- La **modification** de ces documents.

  On peut modifier les documents RDF du forum ou de la base de données d'entretiens capitalisés pour préciser des informations ou corriger d'éventuelles erreurs.

  Une option permettra d'exporter les documents RDF/XML des enquêtes téléphoniques au format SAS.

Le prototype à implémenter servira de base à une application plus robuste multi-poste. Il m'a donc été demandé de fournir une documentation technique complète sur l'application (en Javadoc) pour que la reprise du code ne soit pas fastidieuse.

Chapitre 4. Types de données

Nous allons décrire dans ce chapitre les différents types de données traitées durant le stage. Pour chaque type, nous établirons : une présentation des sources avec un aperçu de leurs structures, les méta-données qu'elles contiennent et enfin les documents que l'application génère (ces derniers utilisent la syntaxe *RDF/XML*, *RDF* est expliqué en détail dans le chapitre 5) . Des diagrammes nous permettront également une vue globale de la structure de chaque modèle.

1. Controverses

Les controverses ont été utilisées pour la réalisation d'un premier prototype en Perl (cf Jakarta ORO), elles concernent un forum de discussion.

Ce forum de discussion est issu d'une base Lotus Domino, il comprend des messages écrits par des internautes et des informations propres au message et à l'internaute; il s'agit principalement du nom et e-mail de l'auteur, du sujet du message, de sa date de création et le message "père" : un message peut être la réponse à un autre message (ex: Message 1 de sujet « Bla bla » père de : Message 2 de sujet « Re : Bla bla ».

1.1. Extrait

Voici un extrait d'un fichier "forum" :

```
publier:  Oui
Date:  01/01/2003 00:52:54
HTTP_Referer:  http://XXXXXXXX.XXX.fr/developpement-durable/frmstakeh.nsf/
               #19863113f421d2f4c1256c4b00347e99?OpenForm
HTTP_User_Agent:  Mozilla/4.0 (compatible; MSIE 5.01; Windows NT)
Sv_Path_Info:  /developpement-durable/frmstakeh.nsf/
               19863113f421d2f4c1256c4b00347e99?CreateDocument
Query_String:  CreateDocument
Remote_Addr:  194.XXX.XXX.34
Request_Method:  POST
Server_Name:  XXXXXXXX.XXX.fr
Server_Protocol:  HTTP/1.1
Server_Software:  Lotus-Domino/5.0.2
NomAuteur:  DYLAN
email:  bobby.dylan@yahoo.fr
Subject:    Réveillon de fin d'année
body:  Si vous ne savez pas quoi faire un premier janvier a minuit,
       reveillonnez donc avec votre ordinateur.
$$Return:  [/developpement-durable/frmstakeh.nsf/Forum?OpenView&count=20
           &EF0516E1A0978834C1256CA000832FC5]
$ConflictAction:  1
export:  oui
$UpdatedBy:  Anonymous,CN=Philippe DEFRETIN/OU=E/O=EDFGDF/C=FR
$Revisions:  01/01/2003 00:52:54
¶

publier:  Oui
Date:  19/12/2002 18:06:18
HTTP_Referer:  http://extranet.edf.fr/developpement-durable/frmstakeh.nsf/
               19863113f421d2f4c1256c4b00347e99?OpenForm
               &ParentUNID=EF0516E1A0978834C1256CA000832FC5
```

```
HTTP_User_Agent:  Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Sv_Path_Info:  /developpement-durable/frmstakeh.nsf/
               19863113f421d2f4c1256c4b00347e99?CreateDocument
               &ParentUNID=EF0516E1A0978834C1256CA000832FC5
Query_String:  CreateDocument&ParentUNID=EF0516E1A0978834C1256CA000832FC5
Remote_Addr:  195.132.14.11
Remote_Host:
Request_Method:  POST
Server_Name:  extranet.edf.fr
Server_Protocol:  HTTP/1.0
Server_Port:  80
Server_Software:  Lotus-Domino/5.0.2
NomAuteur:  Stef le lutin
email: steffy.lutin@edf.fr
Subject:  Re : Réveillon de fin d'année
body:  Oui c'est une excellente idée... merci et bonne année!
$$Return:  [/developpement-durable/frmstakeh.nsf/Forum?OpenView&count=20
           &D160C84EED757B67C1256CA4004C5629]
$ConflictAction:  1
export:  oui
$UpdatedBy:  Anonymous,CN=Philippe DEFRETIN/OU=E/O=EDFGDF/C=FR,
             CN=Stephane COLUCCI/OU=E/O=EDFGDF/C=FR
$Revisions:  02/01/2003 18:06:18,03/01/2003 09:44:29
¶
```

Le forum est donc une succession de messages de cette forme séparée par un caractère spécial. Les informations qui nous intéressent sont les suivantes : le nom de l'auteur du message, son "e-mail", le sujet du message, le corps du message (le message lui-même), la date et l'heure de la création du message. Il nous est également nécessaire de posséder un identifiant de ce message et finalement de connaître l'identifiant de son père..

1.2. Méta-données et Expressions

Pour récupérer les données que l'on a jugées intéressantes, on utilise les expressions *Perl* suivantes :

nom = /^NomAuteur: **(.*)**/

ici “DYLAN"

e-mail = /^email: **(.*)**/

ici “bobby.dylan@yahoo.fr"

sujet = /^Subject : **(.*)**/

ici “Réveillon de fin d'année"

message = /^body: **(.*)**/

et les lignes suivantes tant que celles-ci ne contiennent pas $$Return.

ici “Si vous ne savez pas quoi faire un premier janvier a minuit, reveillonnez donc avec votre ordinateur."

id (identifiant du message) = /$$Return: [.]+ **([A-Z0-9]+)**$/

ici “EF0516E1A0978834C1256CA000832FC5"

parent_id (identifiant du message parent) = /^QueryString : [.]+**([A-Z0-9]+)**$/

ici il n'a pas de père..

date = /$Revisions: **([0-9][0-9]).([0-9][0-9]).([0-9][0-9][0-9][0-9])**/

avec une inversion pour obtenir une notation année-mois-jour.

ici “2003-01-01 (notation US)"

heure = /$Revisions: [.]{2}.[.]{2}.[.]{4} **([0-9][0-9]):([0-9][0-9]):([0-9][0-9])**/

ici “ 00:52:54 “

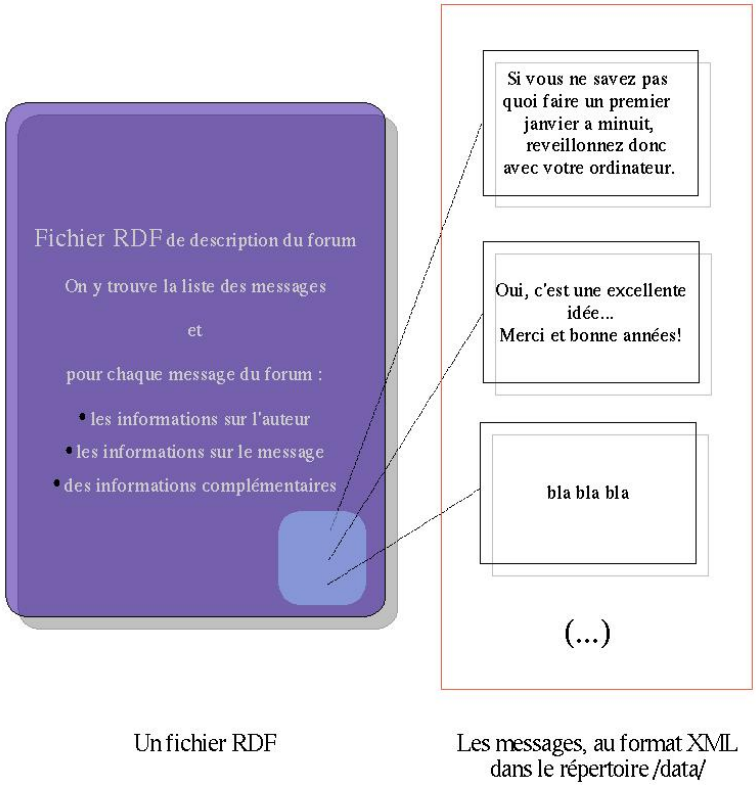
Après la récupération de toutes ces informations, il sera possible de créer les documents RDF et XML.

1.3. Documents générés

La structuration engendre deux types de fichiers :

1. Les fichiers XML correspondant aux messages du forum
2. Un fichier de méta-données décrivant les fichiers précédents.

Figure 4.1. Les deux types de documents générés et les liens les reliant



Le fichier des méta-données:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:pack_metadonnees="file:///schema_p00f/pack_metadonnees.rdf#"
 xmlns:forum="file:///schema_p00f/forum.rdf#"
 xmlns:personne="file:///schema_p00f/personne.rdf#" >
>

  <forum:message rdf:ID="file:///ex_forum/data/id_EF0516E1A0978834C1256CA000832FC5.xml">
    <forum:ref>/ex_forum/data/id_EF0516E1A0978834C1256CA000832FC5.xml</forum:ref>
    <forum:id>EF0516E1A0978834C1256CA000832FC5</forum:id>
    <forum:auteur>
      <personne:personne>
        <personne:nom>DYLAN</personne:nom>
        <personne:email>bobby.dylan@yahoo.fr</personne:email>
      </personne:personne>
    </forum:auteur>
    <forum:sujet>Reveillon de fin d'année</forum:sujet>
```

```
  </forum:message>

  <forum:message rdf:ID="file:///ex_forum/data/id_D160C84EED757B67C1256CA4004C5629.xml">
    <forum:ref>/ex_forum/data/id_D160C84EED757B67C1256CA4004C5629.xml</forum:ref>
    <forum:id>D160C84EED757B67C1256CA4004C5629</forum:id>
    <forum:parent_id>EF0516E1A0978834C1256CA000832FC5</forum:parent_id>
    <forum:auteur>
      <personne:personne>
        <personne:nom>Stef le lutin</personne:nom>
        <personne:email>steffy.lutin@edf.fr</personne:email>
      </personne:personne>
    </forum:auteur>
    <forum:sujet>Re : Réveillon de fin d'année</forum:sujet>
  </forum:message>

(...)

<!-- la ressoure "forum" qui regroupe tous les messages -->

  <forum:forum rdf:ID="file:///ex_forum/ex_forum.rdf">
    <forum:comporte>
      <forum:BagOfMessages>
        <rdf:_1 rdf:resource="#file:///ex_forum/data/id_EF0516E1A0978834C1256CA000832FC5.xml"/>
        <rdf:_2 rdf:resource="#file:///ex_forum/data/id_D160C84EED757B67C1256CA4004C5629.xml"/>
        <rdf:_3 rdf:resource="#file:///ex_forum/data/id_008515BDB9D5EDEBC1256C930031D6FC.xml"/>
      (...)
      </forum:BagOfMessages>
    </forum:comporte>
  </forum>

</rdf:RDF>
```

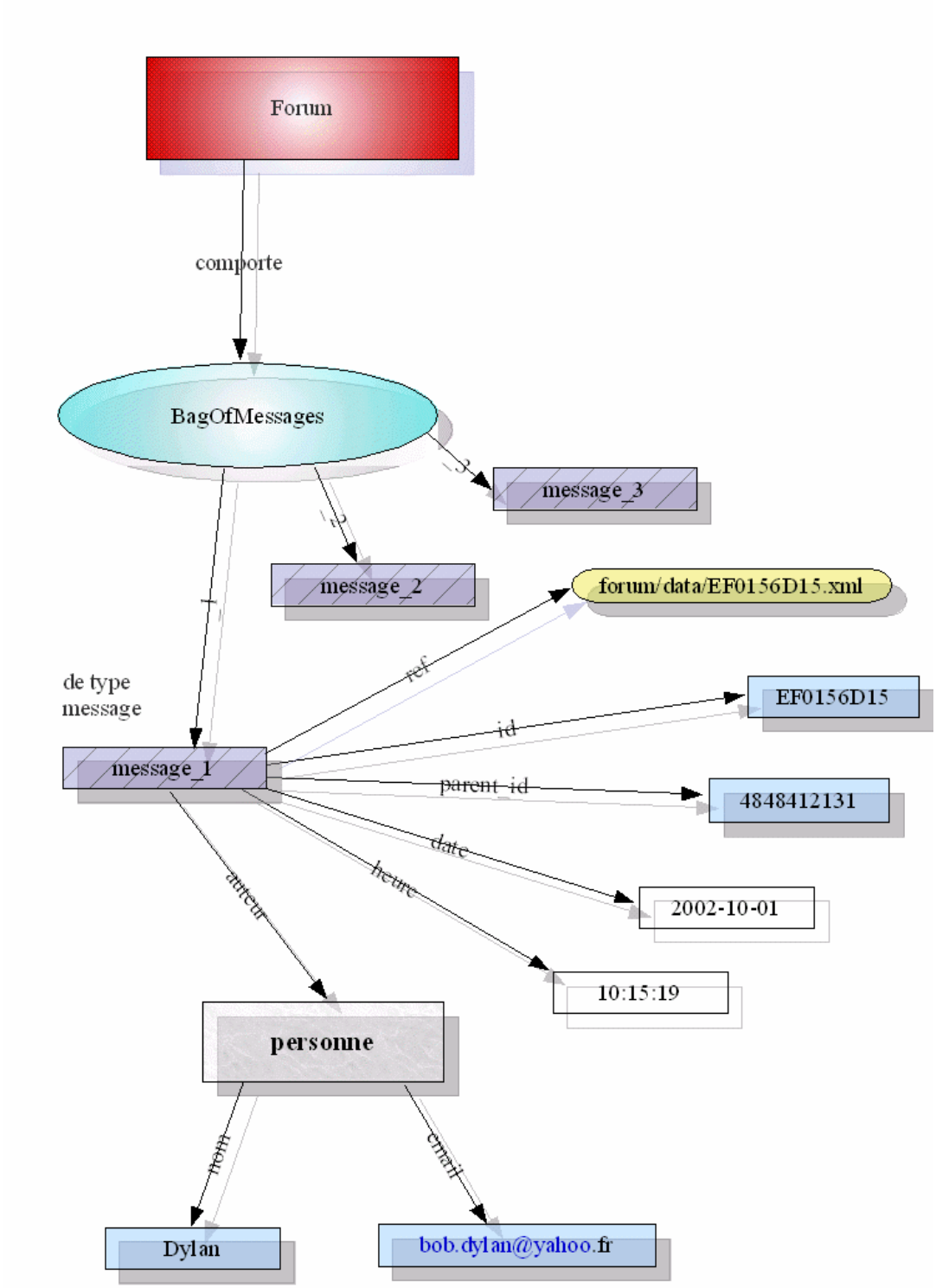
et les fichiers données associés (ici, on a le fichier associé au 1er message, le fichier “id_EF0516E1A0978834C1256CA000832FC5.xml")

```
<?xml version="1.0" encoding="iso-8859-1"?>
<document id="id_EF0516E1A0978834C1256CA000832FC5"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="file:///schema_p00f/document.xsd"
>

  <p id="id_EF0516E1A0978834C1256CA000832FC5_p1">
    Si vous ne savez pas quoi faire un premier janvier a minuit,
    reveillonnez donc avec votre ordinateur.
  </p>
</document>
```

Ce document correspond au premier message issu du forum. Il ne comprend qu'un seul paragraphe (le message du forum).

1.4. Schéma de la structure RDF

Figure 4.2. RDF



2. Base de données d'Entretiens capitalisés

La base de données "d'Entretiens" capitalisés (ou Dixit) est une base comprenant les descriptifs des études et des entretiens menés par des fournisseurs selon un ensemble de critères définis.

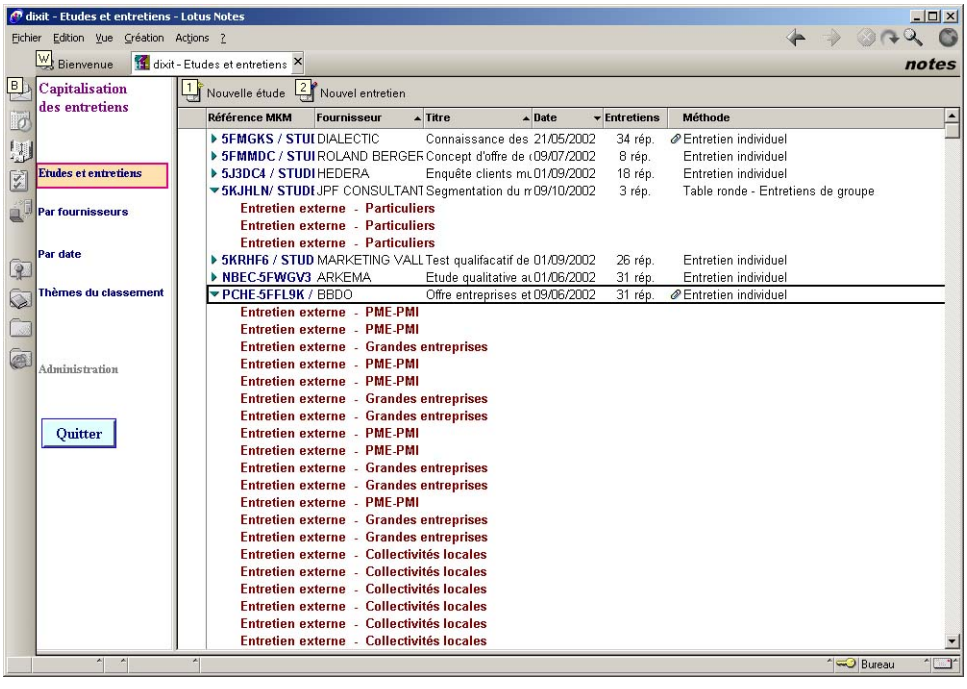
2.1. Extraits

La base de données "Dixit" est stockée dans une base de données "Objets" de type Lotus Notes. Cette base comprend des méta-informations sur les études, les guides d'entretiens (décrivant la manière dont l'entretien se déroule, les sujets abordés, les questions à poser,...) et les entretiens proprement dits. Tous les documents sont de type MS Office (principalement MS Word).

Une étude est composée de plusieurs entretiens.

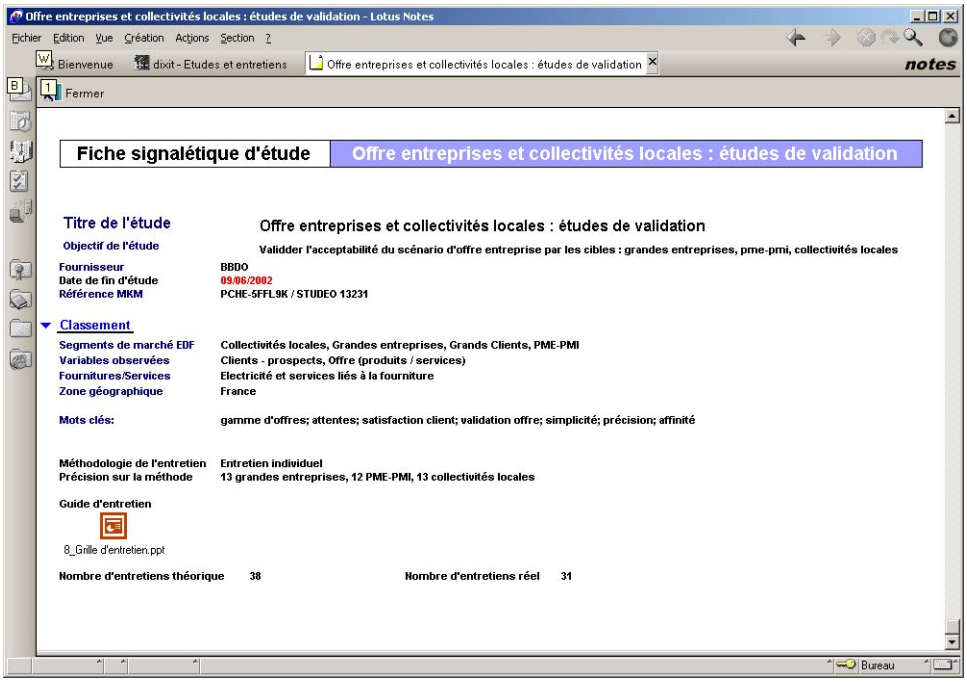
L'accès aux données contenues dans la base se fait par l'intermédiaire de l'interface Notes. Nous voyons sur l'écran suivant l'organisation de la base : les études sont en Bleu et les entretiens qui lui sont rattachés sont en rouge.

Figure 4.3. La base Dixit



Une étude comprend une fiche signalétique et éventuellement un guide d'entretiens (au format MS Word ou PowerPoint).

Figure 4.4. Une étude



"Notes" permet une extraction de la base en format ASCII par l'intermédiaire de la fonction **Exporter** du menu **Fichier**. L'exportation d'une base dixit se présente sous la forme suivante :

```
$FILE:
Fournisseur:  DIALECTIC
ReferenceMKM:  5FMGKS / STUDEO 13186
demographie:  Particuliers
MarcheAff:  Professionnels
VariablesObserveesAff:  Canal de commercialisation - relation clientèle,Clients - prospects,
                        Concurrence,Offre (produits / services)
FournituresAff:  Aures services,Autres utilités,Electricité et services liés à la fourniture,
                 Gaz et services liés à la fourniture
ZoneGeoAff:  France
MotsCles:  segmentation,typologie client,comportement client,création entreprise,
           créateur entreprise,attentes,client,offres hors énergie,relation client,
           image/perception concurrent,comparaison concurrent,financement bancaire,
           diagnostic installation,service conseil,dépannage,maintenance,banques,assurances
valmeth:  1 Entretien individuel
theorique:  34
Doc_Type:  Fiche signalétique d'étude
Intitule:  Connaissance des clients professionnels créateurs d'entreprise
ObjectifEtude:  Cette étude a pour objectif de connaître :
- les profils et les motivations des professionnels créateurs d'entreprise
- les étapes de la création d'une activité professionnelle
- les attentes en matière d'offre énergétique
- leur perception des fournisseurs d'énergie
- leur intérêt à une offre non énergétique d'edf

f_JDateFinEtude:  21
f_MDateFinEtude:  05
f_ADateFinEtude:  2002
ValeurMethod:  Entretien individuel
titresegm:  Segments de marché EDF\Professionnels
titrevarobs:  Variables observées\Canal de commercialisation - relation clientèle,
              Variables observées\Clients - prospects,Variables observées\Concurrence,
              Variables observées\Offre (produits / services)
```

```
titrefourn:  Fournitures / Services\Aures services,Fournitures / Services\Autres utilités,
             Fournitures / Services\Electricité et services liés à la fourniture,
             Fournitures / Services\Gaz et services liés à la fourniture
titrezonegeo:  Zones géographiques\France
classement:  Segments de marché EDF\Professionnels,
             Variables observées\Canal de commercialisation - relation clientèle,
             Variables observées\Clients - prospects,Variables observées\Concurrence,
             Variables observées\Offre (produits / services),
             Fournitures / Services\Aures services,Fournitures / Services\Autres utilités,
             Fournitures / Services\Electricité et services liés à la fourniture,
             Fournitures / Services\Gaz et services liés à la fourniture,
             Zones géographiques\France
f_DateFinEtude:  21/05/2002
$UpdatedBy:  CN=Pascale CHERON/OU=DIRDEV/O=EDFGDF/C=FR
$Revisions:  09/07/2003 11:40:56
¶

$FILE:
Intitule:  Connaissance des clients professionnels créateurs d'entreprise
f_DateFinEtude:  21/05/2002
ReferenceMKM:  5FMGKS / STUDEO 13186
valmeth:  1 Entretien individuel
valmetbis:  1 Entretien individuel
intext:  Externe
demographie:  Particuliers
MarcheAff:  Professionnels
Marcheseg:  Professionnels
sexe1:
sexbis:
age1:
agbis:
csp:
cspbis:
nombre1:
effectif1:
stat:
maison:
chiffre_1:
chauffage:
energie1:
tarif1:
codepostal1:
entreprise:
sexe2:  homme
age2:  37
fonction2:  gérant
codeNAF:
nombre2:  3
effectif:
chiffre:
process:
energie2:  électricité,gaz
tarif2:  45kW
codepostal2:  94400
sexe:
age:
fonction:
codepostal:
Doc_Type:  Fiche signalétique d'entretien
ValeurMethod:  Entretien individuel
logtab:  1
yn:  Externe
segment:  Professionnels
$UpdatedBy:  CN=Pascale CHERON/OU=DIRDEV/O=EDFGDF/C=FR
¶
(...)
```

Le fichier extrait commence par la description de l'étude puis ensuite vient la description de chaque entretien de l'étude.

Ici par exemple, l'étude est fourni par "DIALECTIC" (Fournisseur) et porte sur "la Connaissance des clients professionnels créateurs d'entreprises" (Intitulé). S'enchaînent ensuite, le descriptif de chaque entretien, ils sont espacés par des caractères séparateurs.

2.2. Méta-données

Avant de lire les méta-données, il est nécessaire de savoir s'il s'agit de la description d'une étude ou de celle d'un entretien :

en comparant les deux types de méta-données on remarque qu'une étude commence obligatoirement par « Fournisseur » et un entretien par « Intitulé ».

A l'aide des expressions *Perl*, nous récupérons les méta-données suivantes :

- Pour une étude :
  - le titre
  - le fournisseur
  - la date
  - la référence
  - MKM
  - la zone géographique
  - les mots-clefs (une liste)
  - les variables (une liste)
  - les fournitures
- Pour un entretien :
  - s'il s'agit d'un entretien « interne EDF » ou non
  - le segment de marché
  - la méthodologie

  Nous aurons ensuite, en fonction du type de l'entretien, différents types de méta-données.

  Si l'entretien concerne des **particuliers non interne EDF**:

  - Des informations sociologiques (une ou deux personnes) :

    - le/les sexe(s) des interviewés
    - le/leurs âge(s)
    - le/leurs csp (catégorie sociaux-professionnelle)
    - le nombre de personnes au foyer
    - le nombre d'enfants de moins de 15 ans

- Des informations sur le logement :

    - si la personne interviewée est propriétaire (si non, elle est locataire)
    - s'il s'agit d'une maison (dans le cas contraire, ce sera un appartement)
    - s'il s'agit du domicile principal (dans la cas contraire, il sera secondaire)

- Des informations sur les énergies utilisées

    - le/les type(s) de chauffage
    - le/les énergie(s)
    - les tarifs / puissance(s)

- Des informations géographiques

    - le code postal

Si l'entretien concerne une **entreprise** :

- Des informations sur l'entreprise

    - son nom (peut être une liste)
    - son code NAF
    - l'effectif
    - le chiffre d'affaires

- Des informations sociologiques (une ou plusieurs personnes)
    - le/les sexe(s) des interviewés
    - le/leurs âge(s)
    - le/leurs fonction(s)
- Des informations sur les énergies utilisées
    - le/les type(s) de process
    - le/les énergie(s)
    - les tarifs / puissance(s)
- Des informations géographiques
    - le code postal

Si l'entretien concerne un **interne EDF** :

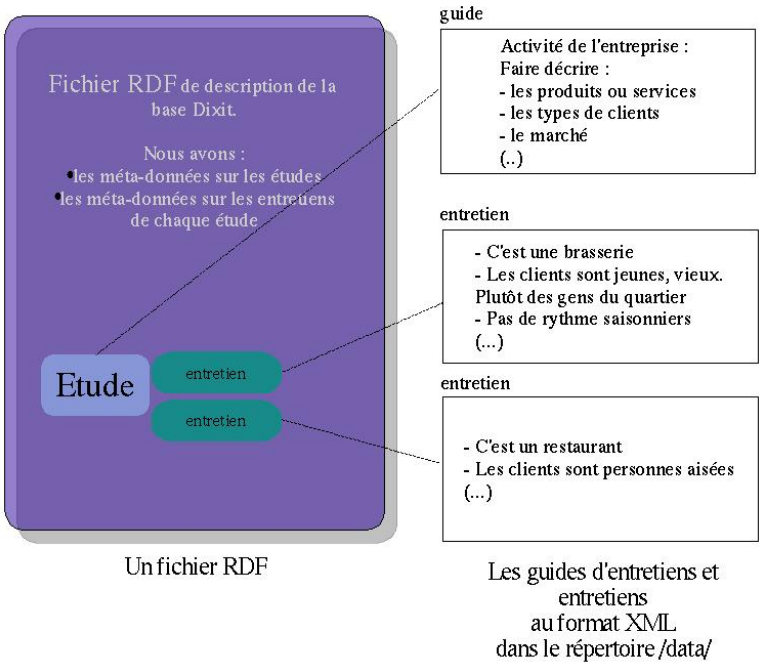
- Des informations socio (une seule personne) :
    - le sexe de l'interviewé
    - son âge
    - son csp (catégorie sociaux-professionnelle)
- Des informations géographiques
    - le code postal

2.3. Document généré

La structuration produit :

- Les guides d'entretien en format XML
- Les entretiens en format XML
- Le fichier RDF qui décrit la base "dixit"

Figure 4.5. Les documents de la base dixit



Voici le document RDF qu'il faudra générer :

```
<?xml version="1.0" encoding="iso-8859-1"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns="file:///schema_p00f/dixit.rdf#"
>

<!-- permière étude -->
<etude rdf:ID="etude_1">

        <fournisseur>DIALECTIC</fournisseur>
        <titre>Connaissance des clients professionnels créateurs d'entreprises</titre>
        <objectif>Cette étude a pour objectif de connaître beaucoup de choses</objectif>
        <ref_MKM>ZZZZZZZ / STUDEO YYYYY</ref_MKM>
        <date>2002-09-21</date>
        <zone_geographique>France</zone_geographique>
        <mots_clefs>
                <rdf:Bag>
                        <rdf:li>segmentation</rdf:li>
                        <rdf:li>typologie client</rdf:li>
                        <rdf:li>comportement client</rdf:li>
```

```
                        <rdf:li>banques</rdf:li>
                        <rdf:li>assurances</rdf:li>
                                                                (...)
                </rdf:Bag>
        </mots_clefs>
        <variables>
                <BagOfVariables>
                        <rdf:li>Clients prospects</rdf:li>
                        <rdf:li>Concurrence</rdf:li>
                        <rdf:li>Offre produits services </rdf:li>
                                                                (...)
                </BagOfVariables>
        </variables>
        <fournitures_services>
                <BagOfFournituresServices>
                        <rdf:li>Electricité</rdf:li>
                        <rdf:li>Gaz</rdf:li>
                                                                (...)
                </BagOfFournituresServices>
        </fournitures_services>
</etude>

<!-- premier entretien de la première étude-->
<entretien rdf:ID="entretien_1_1">

        <interne_EDF>false</interne_EDF>
        <segment_de_marche>Professionnels</segment_de_marche>
        <methodologie>Entretien individuel</methodologie>

        <donnees_socio>
                <donnees>
                <pose_a>
                <personne rdf:ID="personne_1_1_1">
                        <sexe>homme</sexe>
                        <age>37</age>
                        <fonction>gérant</fonction>
                        </personne>
                </pose_a>
                <nombre_de_personnes>3</nombre_de_personnes>
                </donnees>
        </donnees_socio>


        <donnees_energies>
                <donneesEnergies>
                <energies>
                <rdf:Bag>
                        <rdf:li>électricité</rdf:li>
                        <rdf:li>gaz</rdf:li>
                </rdf:Bag>
                </energies>
                <puissances_tarifs>
                <rdf:Bag>
                        <rdf:li>45kW</rdf:li>
                </rdf:Bag>
                </puissances_tarifs>
                </donneesEnergies>
        </donnees_energies>
</entretien>

<entretien rdf:ID="entretien_1_2">
        <interne_EDF>false</interne_EDF>
        <segment_de_marche>Professionnels</segment_de_marche>
        <methodologie>Entretien individuel</methodologie>
        <donnees_socio>
                <donnees>
                <pose_a>
                <personne rdf:ID="personne_1_2_1">
                        <sexe>homme</sexe>
                        <age>31</age>
                        <fonction>gérant</fonction>
                </personne>
                </pose_a>
(...)
</entretien>
(...)
<!-- fin du dernier entretien de l'étude 1, on va faire le lien entre l'étude "1"
        et ces entretiens-->

<etude rdf:about="#etude_1">
        <comporte>
                <BagOfEntretiens>
                        <rdf:li rdf:resource="#entretien_1_1"/>
                        <rdf:li rdf:resource="#entretien_1_2"/>
                        <rdf:li rdf:resource="#entretien_1_3"/>
```

```
(...)
                </BagOfEntretiens>
        </comporte>
</etude>
(...)
<!--la base DIXIT-->

<base_dixit rdf:ID="ex_dixit">
        <est_compose_de>
                <BagOfEtudes>
                        <rdf:li rdf:resource="#etude_1"/>
                        <rdf:li rdf:resource="#etude_2"/>
(...)
                </BagOfEtudes>
        </est_compose_de>
</base_dixit>

</rdf:RDF>
```

Note

Ni le guide d'entretien de l'étude, ni les entretiens bruts de chaque étude ne sont récupérables à partir de cette exportation. Il est donc nécessaire de les extraire de la base de données un par un et les placer en format "txt" dans un répertoire. Le transformateur se chargera de les convertir en XML.

2.4. Schéma de la structure RDF

Figure 4.6. Schéma du modèle RDF d'une base dixit (composé d'études)

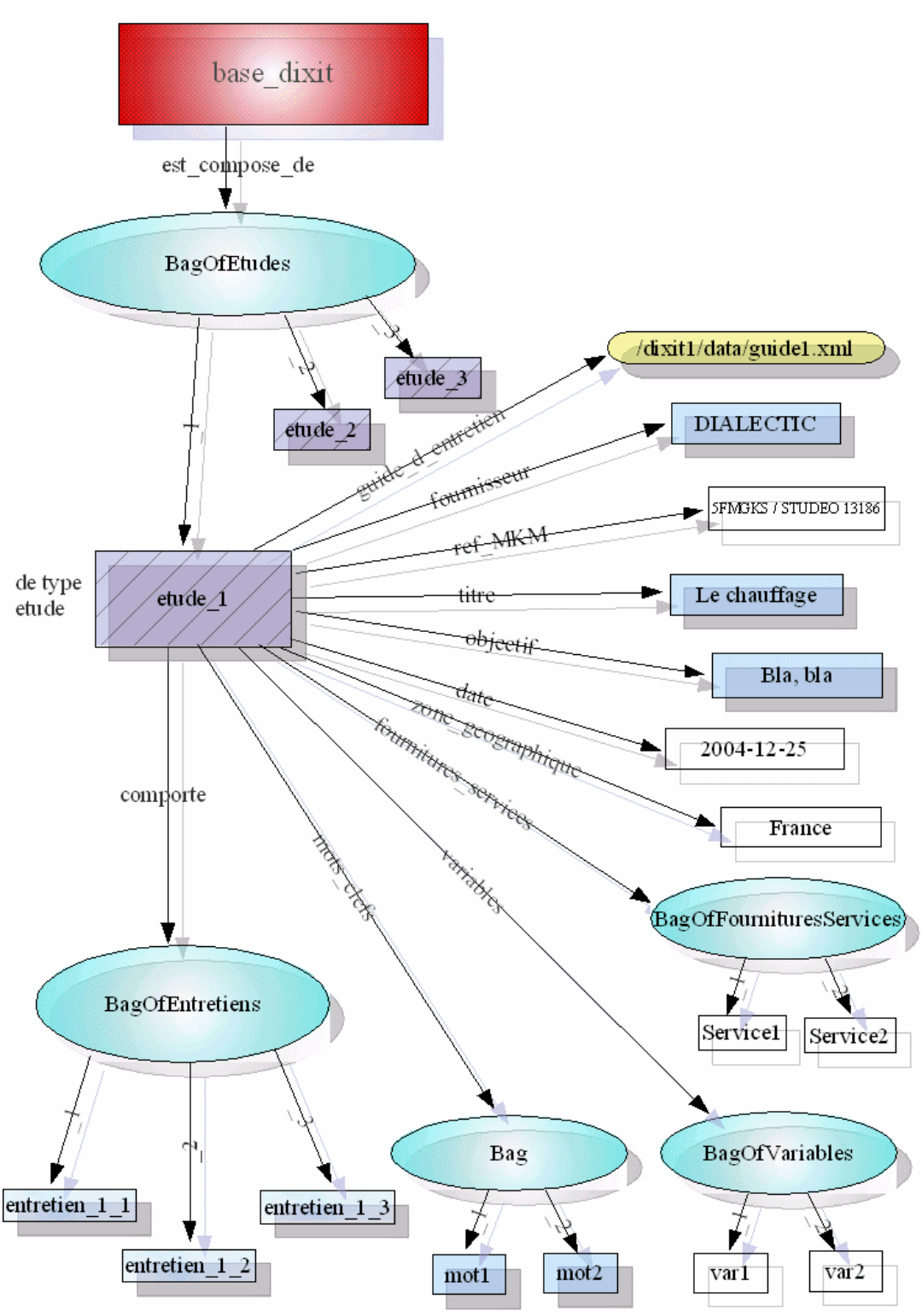
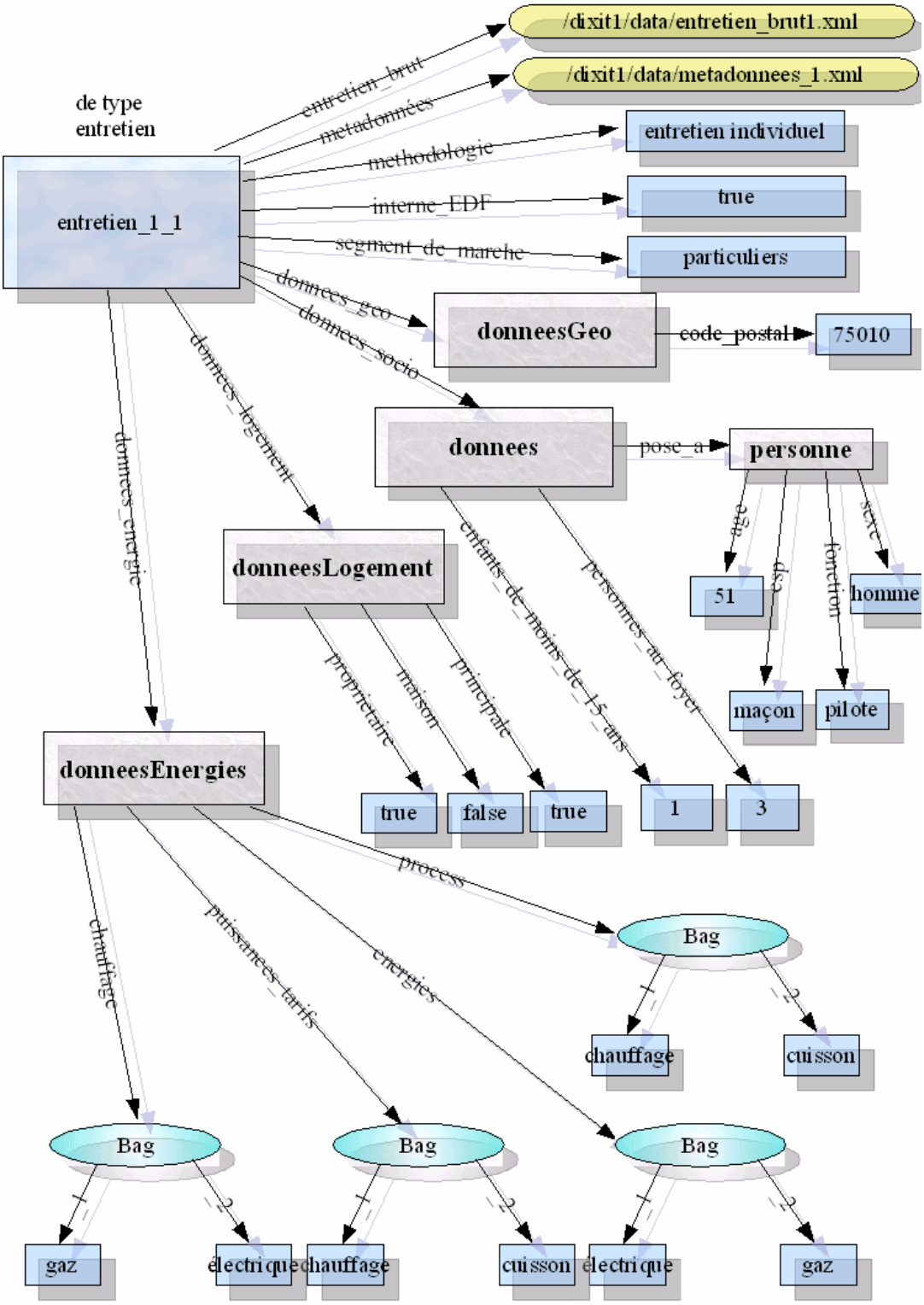


Figure 4.7. Schéma du modèle RDF d'un entretien



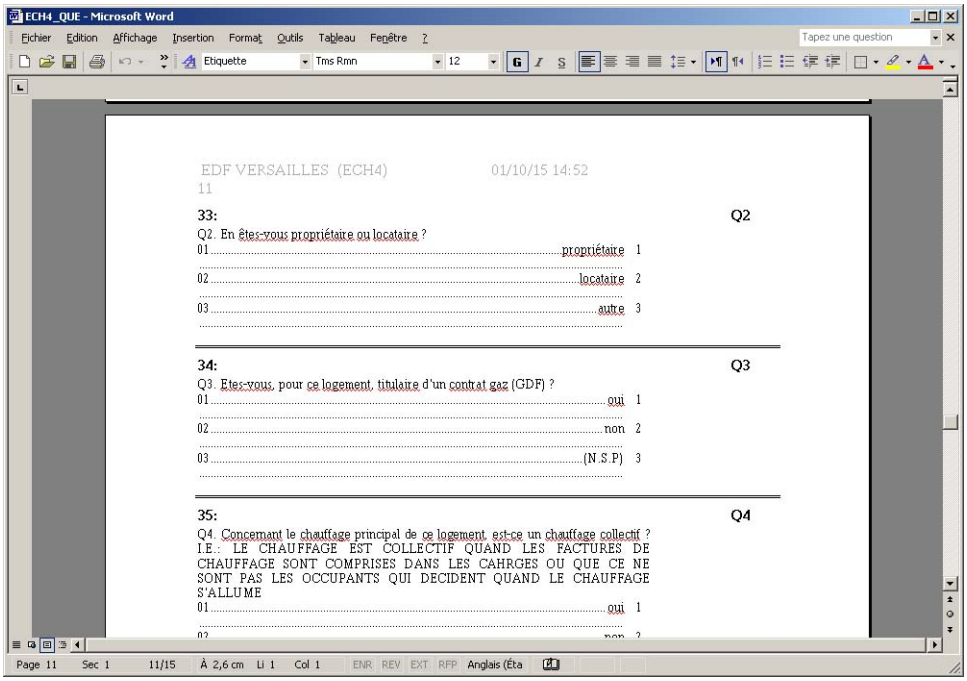
3. Enquêtes Téléphoniques

C'est le département Marketing qui commande des enquêtes téléphoniques ou CATI [9] à une entreprise spécialisée; leur sont fournies des recommandations comprenant l'échantillon de la population à interviewer et les questions à poser. Une fois l'enquête réalisée, le département reçoit de la part du prestataire un fichier questionnaire et un fichier de données contenant le résultat des études (les réponses des interviewés).

3.1. Extraits

Nous allons voir à présent comment ce présente le questionnaire reçu; ce fichier est généré par un logiciel d'interview (ici MV2 Maxiphone Interviewer).

Figure 4.8. Questionnaire



Celui-ci comporte :

1. Le *numéro* de la question précède un *code question* correspondant aux codes donnés par le département Marketing.

9CATI : Computer Assisted Telephon Interview

2. Le *code question* ainsi que la question posée à l'interviewé.

3. Il arrive de rencontrer une *consigne* : une aide à l'interviewer; ces consignes sont précédés par le code "I.E.:"

4. Enfin, les *choix possibles* pour les questions fermées, ainsi que la *signification* de ces choix.

Le fichier de données est un fichier Access; on y trouve le résultat des interviews : le titre des colonnes correspond aux codes questions et chaque ligne contient les réponses données par une (1) personne.

Ces données peuvent être des informations personnelles (nom, adresse, etc.), les réponses aux questions ouvertes ou celles des questions fermées. Dans ce dernier cas, nous n'avons juste que le *code réponse* correspondant au choix; afin de connaître la signification de celui-ci, il est nécessaire de consulter le questionnaire.

Figure 4.9. Table Access

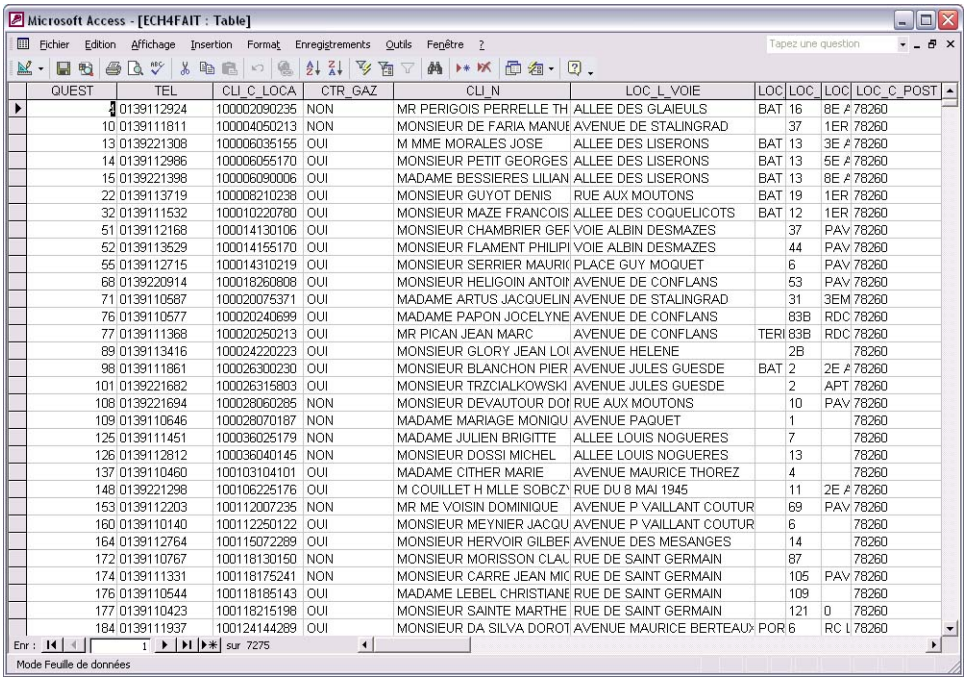
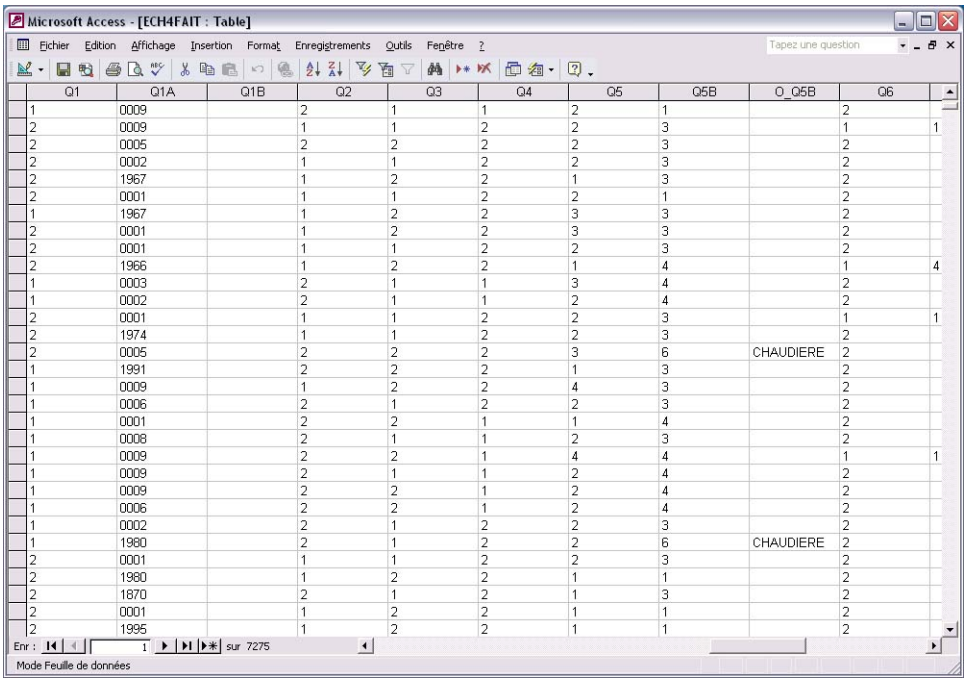


Figure 4.10. Table Access (Questions Q1, Q2, ...)



Q1	Q1A	Q1B	Q2	Q3	Q4	Q5	Q5B	O_Q5B	Q6	
1	0009		2	1	1	2	1		2	
2	0009		1	1	2	2	3		1	1
2	0005		2	2	2	2	3		2	
2	0002		1	1	2	2	3		2	
2	1967		1	2	2	1	3		2	
2	0001		1	1	2	2	1		2	
1	1967		1	2	2	3	3		2	
2	0001		1	2	2	3	3		2	
2	0001		1	1	2	2	3		2	
2	1966		1	2	2	1	4		1	4
1	0003		2	1	1	3	4		2	
1	0002		2	1	1	2	4		2	
2	0001		1	1	2	2	3		1	1
2	1974		1	1	2	2	3		2	
2	0005		2	2	2	3	6	CHAUDIERE	2	
1	1991		2	2	2	1	3		2	
1	0009		1	2	2	4	3		2	
1	0006		2	1	2	2	3		2	
1	0001		2	2	1	1	4		2	
1	0008		2	1	1	2	3		2	
1	0009		2	2	1	4	4		1	1
1	0009		2	1	1	2	4		2	
1	0009		2	2	1	2	4		2	
1	0006		2	2	1	2	4		2	
1	0002		2	1	2	2	3		2	
1	1980		2	1	2	2	6	CHAUDIERE	2	
2	0001		1	1	2	2	3		2	
2	1980		1	2	2	1	1		2	
2	1870		2	1	2	1	3		2	
2	0001		1	2	2	1	1		2	
2	1995		1	2	2	1	1		2	

3.2. Méta-données / Lecture d'un fichier CSV

Nous allons lire le fichier questionnaire et extraire les méta-données suivantes :

1. Ligne 1 de chaque bloc de questions : nous sert de ligne référence.
2. Ligne 2

   - Si la ligne est de la forme /([^\:])\:(.+)/ (c'est-à-dire qu'elle contient une expression suivie de deux points)

     Nous sommes alors en présence d'une *donnée importée* provenant d'une base EDF. Il nous faut alors connaître le code de cette donnée

     - Si cette ligne contient l'expression /\(code/ ou /^[^\(\)]+$/

       Alors la *donnée importée* est de *code* /^([^\(\)]+)/ (en prenant comme ligne lue la valeur du $1 de l'expression précédente)

     - Sinon la *donnée importée* est de *code* / \(([^\)]+)\)/

   - Sinon il s'agit d'une *question* possédant un intitulé

     A partir de l'expression /([^\.]+)\./,

on récupère :

le *code* question : $1

L'*intitulé* de la question : $' (les caractères suivants l'expression précédente)

3. Ligne 3

   Il peut y avoir une remarque (consigne) = /i\.e\.\:(.+)/i

4. Ligne 4 et suivantes

   Nous pouvons avoir les possibilités de réponses.

   Si l'expression /^[\\d]+[\\t]+([^\\t]+)[\\t]([^\\t]+)/ est vraie,

   alors pour la *possibilité* $2 nous avons la *signification* $1.

Afin de structurer le fichier Access, il est nécessaire de passer par un format intermédiaire qui pourra être lu facilement.

Le format CSV offre la particularité de pouvoir contenir des tableaux volumineux et en même temps d'avoir une structure simple; on peut résumer la structure d'un fichier CSV en deux points :

- Les champs sont séparés par un *caractère séparateur* ("," ou ";")
- La première ligne contient les labels de chaque colonne; ces labels sont séparés par le *caractère séparateur*; et les lignes suivantes sont les lignes données.

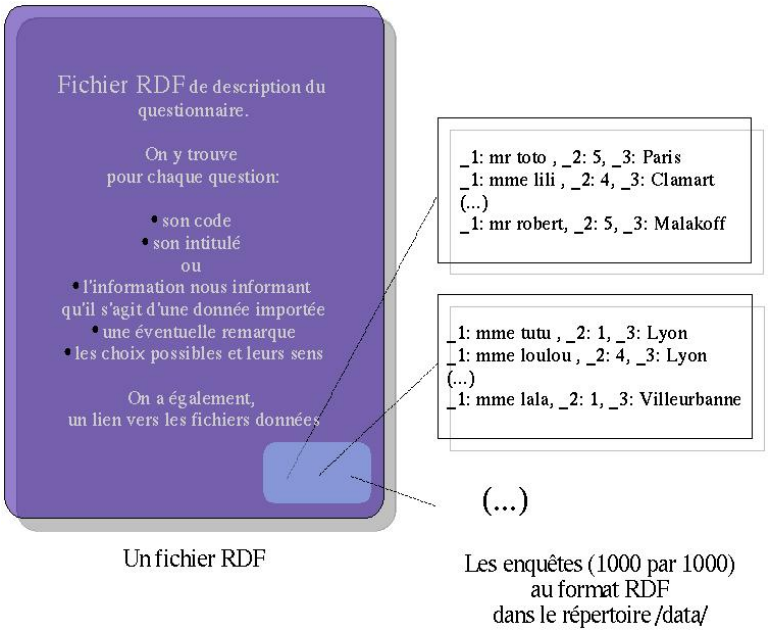
Avant de structurer les enquêtes CATI, il est donc nécessaire d'**exporter** le fichier Access en fichier **CSV** exploitable par l'application.

3.3. Documents générés

La structuration des deux fichiers sources génère deux types de fichiers : un fichier principal contenant la structure du questionnaire et des fichiers données.

Le fichier principal comprend l'intégralité des méta-informations contenues dans le *questionnaire* (questions, codes, possibilités de réponses, etc.) et en plus un lien vers les *fichiers données*. Ces fichiers contiennent 1000 entretiens chacun, on y trouve les réponses pour les questions ouvertes les codes de réponses pour les questions fermées et des liens vers la question.

Figure 4.11. Les deux types de documents générés et les liens les reliant



3.4. Schéma de la structure RDF

Figure 4.12. Schéma du modèle : une enquête

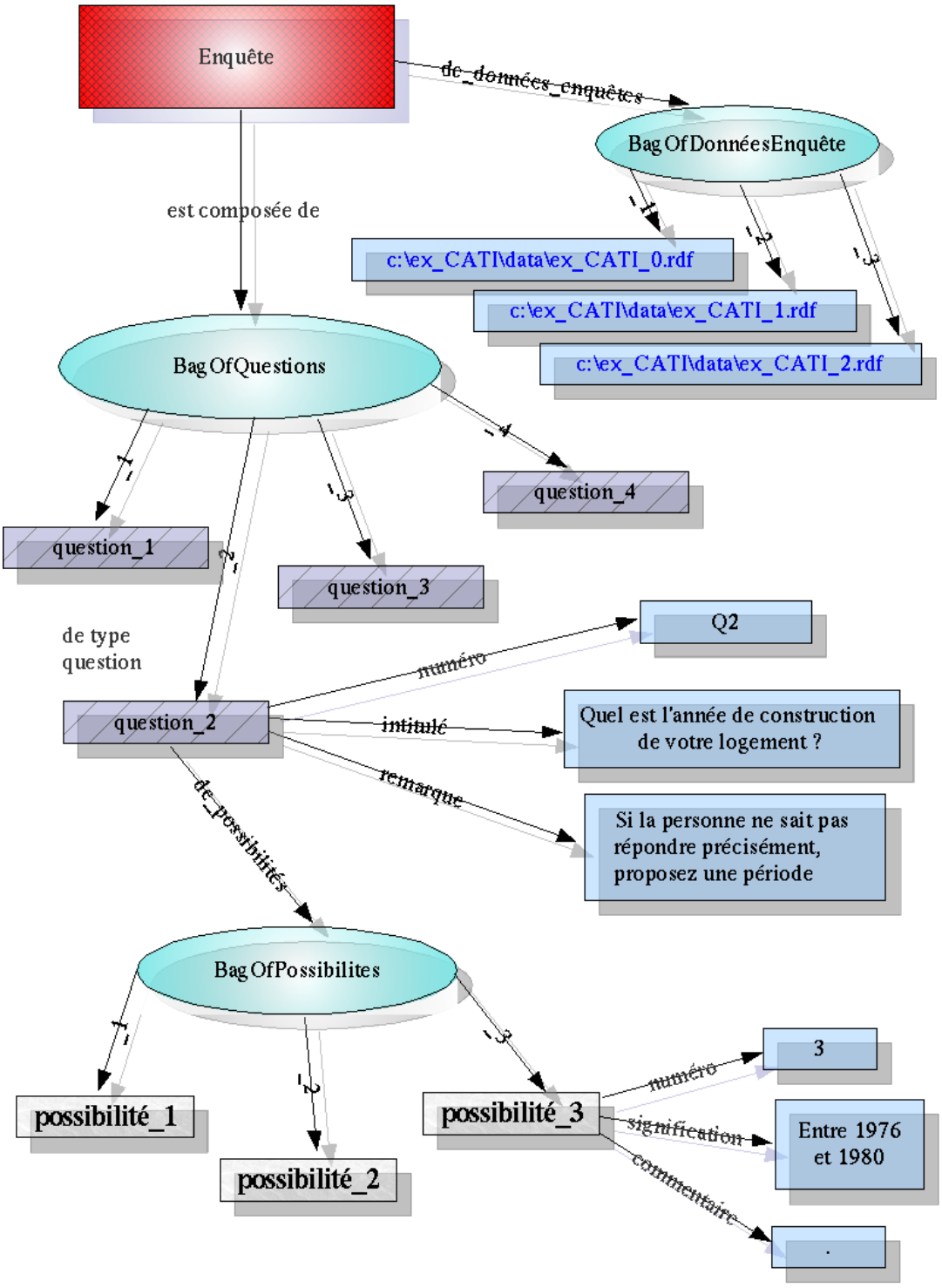
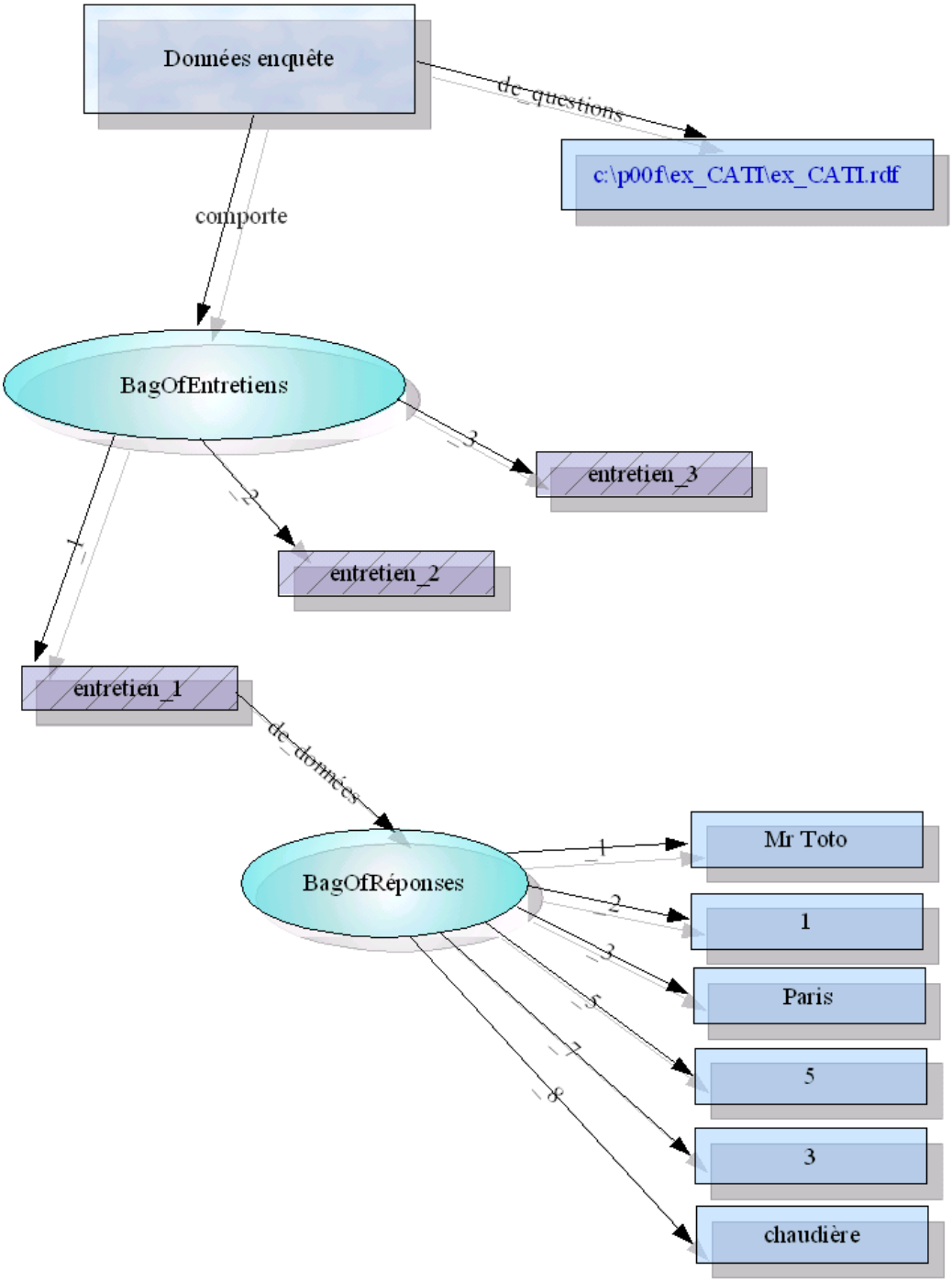


Figure 4.13. Schéma du modèle : les données enquête



Chapitre 5. Langages de description

Dans cette partie, nous aborderons les différentes technologies servant de *support* à *l'information*. En effet, les "informations" jouent un rôle prépondérant, en ce sens qu'elles sont à la base de toute communication entre les êtres vivants. Celles-ci doivent être structurées pour être compréhensibles et réutilisables. L'accès à celles-ci doit se faire de la manière la plus simple possible, les langages de description servent dans ce projet à stocker les "informations".

1. XML

XML (eXtensible Markup Language soit Langage à balises étendues, ou Langage à balises extensibles) est en quelque sorte un langage HTML amélioré permettant de définir de nouvelles balises. Il s'agit d'un langage permettant de mettre en forme des documents grâce à des balises (markup).

XML a été conçu pour des documents arbitrairement complexes, tout en s’appuyant sur cinq grands principes simples et clairs :

- Lisibilité à la fois par les machines et par les utilisateurs,
- Définition sans ambiguïté du contenu d’un document,
- Définition sans ambiguïté de la structure d’un document,
- Séparation entre documents et relations entre documents,
- Séparation entre structure du document et présentation du document.

Contrairement à HTML, qui est considéré comme un langage défini et figé (avec un nombre de balises limité), XML peut être considéré comme un métalangage permettant de définir d'autres langages, c'est-à-dire définir de nouvelles balises permettant de décrire la présentation d'un texte. La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. Il va permettre de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir.

De plus, les balises XML décrivent le contenu plutôt que la présentation, ce qui permet par exemple d'afficher un même document sur des applications ou des périphériques différents sans pour autant nécessiter de créer autant de versions du document que l'on nécessite de représentations. Un document XML peut être présenté sous la forme d'un arbre et son contenu est exploré en parcourant les branches de celle-ci.

XML a été mis au point par le XML Working Group sous l'égide du World Wide Web Consortium (W3C, l'organisme chargé de standardiser les évolutions du Web) dès 1996. Depuis le 10 février 1998, les spécifications XML 1.0 ont été reconnues comme recommandations par le W3C, ce qui en fait un langage reconnu. (Tous les documents liés à la norme XML sont consultables et téléchargeables sur le site Web du W3C, http://www.w3c.org/XML/ [http://www.w3c.org/XML]).

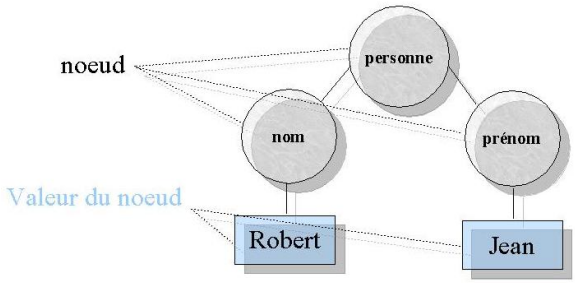
XML est un sous ensemble de SGML (Standard Generalized Markup Language), défini par le standard ISO8879 en 1986, utilisé dans le milieu de la Gestion Électronique Documentaire (GED). XML reprend la majeure partie des fonctionnalités de SGML, il s'agit donc d'une simplification de SGML afin de le rendre utilisable sur le Web.

Voici l'exemple d'un document XML contenant des informations sur une personne :

```
<?xml version='1.0'?>
<personne>
  <nom>Robert</nom>
  <prénom>Jean</prénom>
</personne>
```

et voici l'arbre correspondant à ce document

Figure 5.1. Arbre XML



Le noeud père est le noeud *personne*, il possède deux fils : le fils droit *nom* (de valeur "Robert") et le fils gauche *prénom* (de valeur Jean).

2. Pourquoi RDF?

Chaque jour, des milliers de nouveaux documents viennent peupler le "web", son volume augmente ainsi de jour en jour. Les méthodes traditionnelles de recherche s'essouflent peu à peu : l'utilisation des index n'est plus suffisante pour analyser convenablement les giga-octets contenus sur la toile. Les moteurs de recherche nous ramènent du *bruit* (documents non-pertinants) et l'utilisateur est contraint de faire le tri de toutes les réponses qui lui sont fournis. Le *Web Sémantique* pourrait être la solution de ce problème. Celui-ci, proposé par le W3C, suggère deux éléments :

- l'adjonction aux contenus informels du Web (page HTML, image, etc.) de connaissances formalisées réutilisables par des traitements automatiques.
- l'élargissement de la recherche : du document textuel à des ressources de tout type (image, vidéo, son, services, etc.)

Deux technologies ont été développées dans cette vision du Web Sémantique : RDF et les Topic Maps[10].

RDF ou Resource Description Framework est donc une insfrastructure proposée par le Consortium W3C. Globalement, on peut dire que RDF permet de décrire "intelligemment" une ressource (une page web, etc.). Cette annotation est effectuée d'une manière externe à la ressource et utilise un vocabulaire particulier (RDF Schéma).

La partie suivante traite de façon plus détaillée et précise la technologie RDF.

3. RDF

RDF a été créé par le W3C dans le but d'automatiser le traitement automatique des ressources disponibles sur le Web.

Le principe est d'affecter des méta-données à la description de ressources.

Les ressources correspondent à tout objet présent sur le Web et accessible par le biais d'un identificateur de ressources : un URI (Uniform Resource identifier).

Cet URI peut être une page Web (HTML), un fichier d'un autre format, d'une collection de fichiers ou de toute autre entité (personnes, sociétés, etc.).

[10]Topic Maps : Cartes Thématiques

Ex :

Le site Web du Limsi --> http://www.limsi.fr/

La page Web du Département CHM --> http://www.limsi.fr/Recherche/CHMdp.html

Le logo du Limsi (image) --> http://www.limsi.fr/Images/logorelief100.gif

1. Définition d'une *méta-donnée* :

   « Donnée décrivant le contexte, le contenu et la structure des documents ainsi que leur gestion dans le temps. » [ISO 15489-1 2001]

   « Donnée qui renseigne sur la nature de certaines autres données et qui permet ainsi leur utilisation pertinente. » [OQLF]

   Une méta-donnée est une donnée à propos d'une autre donnée.

   Ex :

   l'auteur d'un message

   la profession d'une personne

2. Le *modèle de donnée* RDF

   Le modèle de donnée élémentaire repose sur trois types d'objets : les ressources, les propriétés et les déclarations.

   Les propriétés représentent des caractéristiques, des aspects, des attributs ou de relations particulières des ressources et en définissent les valeurs descriptives.

Tableau 5.1. Exemples de propriétés

sujet	propriété	valeur
message_0001.xml	date de création	12 juillet 2003
message_0001.xml	auteur	Jean Robert

Ici, nous voyons les deux déclarations composées chacune d'un sujet (la ressource), d'un prédicat (la propriété) et d'un objet (la valeur); déclarations matérialisant les dépendances et les composants d'une relation RDF.

Cette description des ressources est effectuée par l'intermédiaire d'éléments RDF décrivant des propriétés et leurs valeurs.

Ex :

Nous avons le message suivant écrit par *Jean Robert* sur un forum de discussion, le fichier est stocké sur le serveur avec comme nom de fichier *message0001.xml* :

"La libéralisation accrue du marché français de l'énergie pourrait se traduire par une hausse des prix de l'électricité et du gaz pour les clients pouvant faire jouer la concurrence, selon des experts du secteur. Cette augmentation des prix à prévoir peut s'expliquer par les besoins d'investissements croissants induits par l'ouverture du marché plus que par la libéralisation elle-même (...)"

Nous avons la déclaration suivante, relative au message précédent:

Le fichier http://forum.edf.fr/message0001.xml *de type* "message", *a comme sujet* "l'ouverture du marché", *a été écrit le* 12 juillet 2003 par Jean Robert.

Nous avons donc, les caractéristiques suivantes, leurs valeurs associées et l'URI de la ressource :

La ressource (URI = "http://forum.edf.fr/message_0001.xml")

de type --> "message"

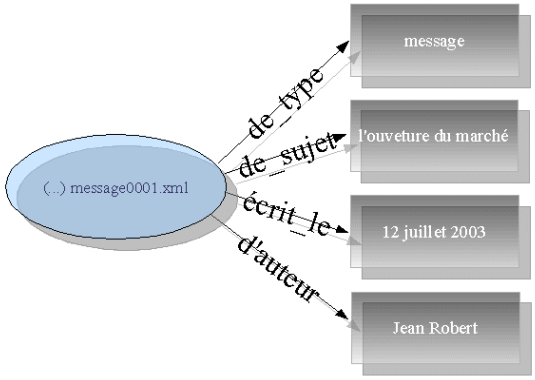
de sujet --> "l'ouverture du marché"

écrit le --> "12 juillet 2003"

d'auteur --> "Jean Robert"

Voici le diagramme de la déclaration RDF correspondant aux déclarations précédentes.

Figure 5.2. Le schéma correspondant



Le code de la déclaration RDF se fait sous la forme d'un triplet <sujet, *prédicat*, objet>.

Ici nous allons avoir les triplets suivants :

<http://forum.edf.fr/message_0001.xml, *type*, "message">

<http://forum.edf.fr/message_0001.xml, *de_sujet*, "l'ouverture du marché">

<http://forum.edf.fr/message_0001.xml, *écrit_le*, "12 juillet 2003">

<http://forum.edf.fr/message_0001.xml, *d_auteur*, "Jean Robert">

3. RDF/XML

Le codage de nos méta-données RDF sera effectué selon le standard XML défini par le consortium *W3C* (http://www.w3c.org/XML).

Voici le code RDF/XML correspondant à la déclaration précédente :

```
1 : <?xml version="1.0" encoding="iso-8859-1"?>

2 : <rdf:RDF
3 :  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4 :  xmlns:vocabulaire="http://p000f.edf.fr/schema/vocabulaire_forum.rdf#"
5 : >

6 :  <rdf:Description rdf:about="http://forum.edf.fr/message_0001.xml">
7 :    <vocabulaire:type>message</vocabulaire:type>
8 :   </rdf:Description>

9 :  <rdf:Description rdf:about="http://forum.edf.fr/message_0001.xml">
10:    <vocabulaire:sujet>l'ouverture du marché</vocabulaire:sujet>
11:  </rdf:Description>

12:  <rdf:Description rdf:about="http://forum.edf.fr/message_0001.xml">
13:    <vocabulaire:ecrit_le>12 juillet 2003</vocabulaire:ecrit_le>
14:  </rdf:Description>

15:  <rdf:Description rdf:about="http://forum.edf.fr/message_0001.xml">
16:    <vocabulaire:auteur>Jean Robert</vocabulaire:auteur>
17:  </rdf:Description>

18: </rdf:RDF>
```

A la ligne 1 nous avons les déclarations relatives à XML (version et type d'encodage)

Ensuite, ligne 2: un fichier RDF/XML commence toujours par la balise <rdf:RDF>

A la ligne 3 et 4, nous avons des déclarations en ce qui concerne les espaces de noms XML (xmlns:XML NameSpace). Sous cette abréviation est spécifiée la création des balises XML spéciales.

Ex, pour "xmlns:rdf", nous allons donc avoir des balises RDF (qui

commenceront par "rdf:")

De la même manière nous allons avoir des balises qui commenceront par "r:" (un espace de nom que nous avons nous-même défini).

La déclaration commence réellement à la ligne 6. Ici, nous avons l'URI de la **ressource** qui va être décrite et à la ligne 7, nous avons la **propriété** "type" et la **valeur** "page HTML". La balise "rdf:Description" est fermée à la ligne 8.

Les lignes 6,7 et 8 correspondent donc à la représentation RDF/XML du triplet:

<http://forum.edf.fr/message_0001.xml, *type*, "message">

De la même manière, les 3 autres triplets ont été mis sous forme RDF/XML et sont sur les lignes :

- 9, 10, 11 pour <http://forum.edf.fr/message_0001.xml, *de_sujet*, "l'ouverture du marché">
- 12, 13, 14 pour <http://forum.edf.fr/message_0001.xml, *ecrit_le*, "12 juillet 2003">
- 15, 16, 17 pour <http://forum.edf.fr/message_0001.xml, *auteur*, "Jean Robert">

Ici nous avons donc rajouté à la ressource **"http://forum.edf.fr/message_0001.xml"** les propriétés une par une. Le mot clef *rdf:about* permet de désigner une ressource.

Dans le code suivant, nous allons déclarer la ressource avec le mot clef *rdf:ID* : (identifiant unique). Les quatre propriétés ne seront plus comme précédemment ajoutées une à une mais associées à la ressource en une seule "passe".

```
1 : <?xml version="1.0" encoding="iso-8859-1"?>

2 : <rdf:RDF
3 :  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4 :  xmlns:vocabulaire="http://p000f.edf.fr/schema/vocabulaire_forum.rdf#"
5 : >

6 :   <rdf:Description rdf:ID="http://forum.edf.fr/message_0001.xml">
7 :     <vocabulaire:type>message</vocabulaire:type>
8 :     <vocabulaire:sujet>l'ouverture du marché</vocabulaire:sujet>
9 :     <vocabulaire:ecrit_le>12 juillet 2003</vocabulaire:ecrit_le>
10:     <vocabulaire:auteur>Jean Robert</vocabulaire:auteur>
11:   </rdf:Description>

12: </rdf:RDF>
```

4. Référence à une ressource

Dans l'assertion RDF (annotation : assertion = triplet) <sujet, *prédicat*, objet>, nous avions eu comme "objet" du prédicat des "valeurs" (par exemple : "page HTML" ou encore "5 avril 2004")

Il est possible d'avoir comme "objet" d'une assertion, une ressource :

Soit la déclaration suivante :

Jean Robert participe aux débats organisés sur un forum de discussion, il possède l'adresse "e-mail" jean.robert@edf.fr et a écrit deux messages :

- le premier (message_0001.xml), dont le sujet était : "l'ouverture du marché" écrit le 12 juillet 2003
- et le second (message_0015.xml) concernant l'énergie nucléaire le 15 juillet 2003

La déclaration précédente du point de vue de la "description des messages" pourrait être représentée de telle manière :

La ressource d'URI "message_0001.xml"

de type --> "message"

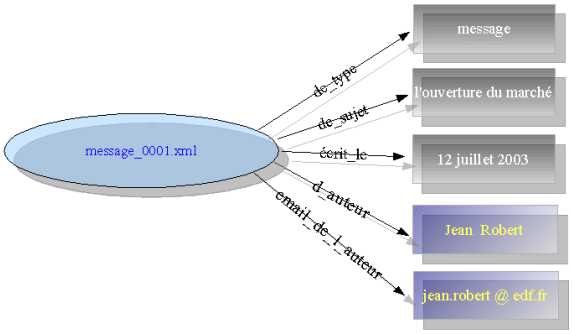
de sujet --> "l'ouverture du marché"

écrit le --> "12 juillet 2003"

d'auteur --> "Jean Robert"

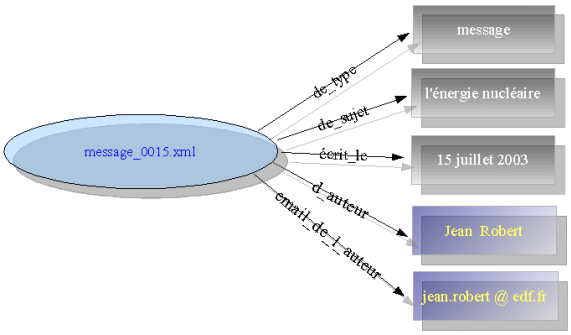
email de l'auteur --> "jean.robert@edf.fr"

Figure 5.3. Schéma d'un message



et

Figure 5.4. Un autre message



La ressource d'URI "message_0015.xml"

de type --> "message"

de sujet --> "l'énergie nucléaire"

écrit le --> "15 juillet 2003"

d'auteur --> "Jean Robert"

email de l'auteur --> "jean.robert@edf.fr"

On remarque que la déclaration de l'auteur est redondante. Créer une ressource "auteur" et relier les oeuvres à celle-ci nous permettent d'éviter la redondance :

La ressource d'URI "auteur001"

de nom --> "Jean Robert"

d'email --> "jean.robert@edf.fr"

La ressource d'URI "message_0001.xml"

est de type --> "message"

de sujet --> "l'ouverture du marché"

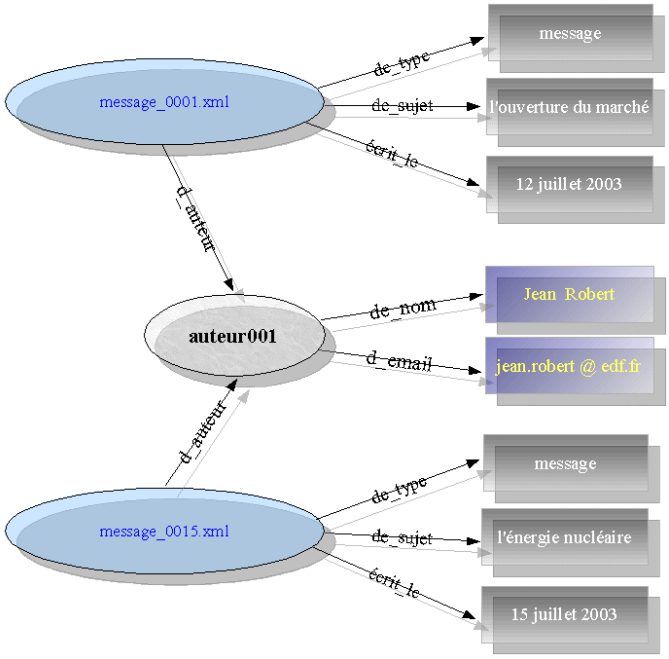
écrit le --> "12 juillet 2003"

d'auteur --> #auteur001

La ressource d'URI "message_0015.xml"

de type --> "message"

Figure 5.5. L'auteur des deux messages est le même



de sujet --> "l'énergie nucléaire"

écrit le --> "15 juillet 2003"

d'auteur --> #auteur001

Il est à noter que dans cet exemple, l'auteur n'est plus une valeur entre guillemets mais une référence à une ressource (l'identifiant de la ressource précédée de #), en RDF/XML, ce lien se traduit par le mot clef *rdf;resource* : rdf:resource="#<nom de la ressource>".

5. Les containers

Les containers en RDF permettent de gérer les collections de valeurs ou de ressources. Chaque élément du "container" est contenu entre des balises *rdf:li*.

- Alternatives

  Il est possible en RDF d'implémenter le choix alternatif:

  ex: La "Situation familiale" d'une femme est, au choix :

- célibataire
- mariée
- pacsée
- en concubinage
- divorcée
- veuve

Ces alternatives seraient codées en RDF/XML de la manière suivante :

```
<rdf:Description rdf:ID="Situation_Familiale">
   <rdf:Alt>
     <rdf:li>célibataire<rdf:li>
     <rdf:li>mariée<rdf:li>
     <rdf:li>pacsée<rdf:li>
     <rdf:li>en concubinage<rdf:li>
     <rdf:li>divorcée<rdf:li>
     <rdf:li>veuve<rdf:li>
   </rdf:Alt>
</rdf:Description>
```

- Emballage (Bag)

  Un "Bag" est une collection d'objets, ils ne sont pas implicitement ordonnés :

  Un forum de discussion est composé de 3 messages :

  - message_001.xml de sujet "l'ouverture du marché" de Jean Robert
  - message_002.xml de sujet "Re : l'ouverture du marché" de Juliette R.
  - message_003.xml de sujet "EDF GDF" de Juliette R.

  La ressource Forum *est composée de* plusieurs messages (dont l'ordre ne nous intéresse pas) :

```
<rdf:Description rdf:ID="forum">
   <vocabulaire:est_composee_de>
     <rdf:Bag>
       <rdf:li rdf:resource="#message001.xml"/>
       <rdf:li rdf:resource="#message002.xml"/>
       <rdf:li rdf:resource="#message003.xml"/>
     </rdf:Bag>
   </vocabulaire:est_compose_de>
</rdf:Description>
```

  Remarque : un "Emballage" peut contenir une suite d'éléments ordonnés comme une Séquence. Il faut alors utiliser les balises *rdf:_n*.

- Sequence (Seq)

  Une séquence, comme son nom l'indique, est une suite d'éléments. L'ordre de ces éléments est donc important et est dicté par leurs classements dans la Séquence.

  Ex : Les messages de Jean Robert, *ordonnés* chronologiquement, sont :

  1er : message0001.xml de sujet "l'ouverture du marché"

  2ème : message0015.xml de sujet "l'énergie nucléaire"

  3ème : message0016.xml de sujet "Re : l'énergie nucléaire"

  Nous aurons :

```
<rdf:Description rdf:ID="forum">
   <vocabulaire:est_composee_de>
     <rdf:Seq>
       <rdf:li rdf:resource="#message0001.xml"/>
       <rdf:li rdf:resource="#message0015.xml"/>
       <rdf:li rdf:resource="#message0016.xml"/>
     </rdf:Seq>
   </vocabulaire:est_composee_de>
</rdf:Description>
```

  ou

```
<rdf:Description rdf:ID="forum">
   <vocabulaire:est_composee_de>
      <rdf:Seq>
         <rdf:_2 rdf:resource="#message0015.xml"/>
         <rdf:_1 rdf:resource="#message0001.xml"/>
         <rdf:_3 rdf:resource="#message0061.xml"/>
      </rdf:Seq>
   </vocabulaire:est_composee_de>
</rdf:Description>
```

  Si les éléments sont mélangés durant la déclaration, la balise *rdf:_n* (avec n > 0) permet de fixer leurs ordres. Ceci fonctionne également pour un Emballage (Bag).

4. RDF Schema

Nous avions vu précédemment la spécification de la syntaxe du langage RDF, on a implicitement parlé de la spécification du vocabulaire RDF : les schémas RDF.

En effet, nous avions utilisé dans les différents exemples RDF : les éléments *Description*, *Bag* ou *li* sont des éléments de ce vocabulaire (il sont tous précédés par "rdf:" : rdf:*Description*, rdf:*Bag* ou rdf:*li*).

Il convient maintenant d'utiliser notre propre vocabulaire, utilisation qui doit être précédée de la déclaration du vocabulaire.

Nous définissons les différentes classes et les différentes propriétés que peuvent posséder celles-ci. Pour cette définition, on utilise :

- rdfs:Class pour définir une nouvelle classe
- rdfs:subClassOf, utilisé pour exprimer le fait qu'une classe hérite d'une autre classe (comme en programmation orientée objet), elle en possède donc toutes les caractéristiques (les propriétés).

Ex : On définit la classe A et la classe B qui descend de la classe A :

```
<rdfs:Class rdf:ID="A">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
  <!--on peut ne pas mettre la ligne précédente, elle est implicite car toute classe
  descend de la classe Resource-->
</rdfs:Class>

<rdfs:Class rdf:ID="B">
  <rdfs:subClassOf rdf:resource="#A"/>
</rdfs:Class>
```

On définit les propriétés avec le vocabulaire suivant :

- rdf:Property pour définir une nouvelle propriété
- rdfs:domain pour associer la propriété à la classe
- rdfs:range pour donner le type de la valeur que peut prendre la propriété

Ex : On définit la propriété pA de la classe A, pB de la classe B :

```
<rdf:Property rdf:ID="pA">
  <rdfs:domain rdf:resource="#A"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:ID="pB">
  <rdfs:domain rdf:resource="#B"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>
```

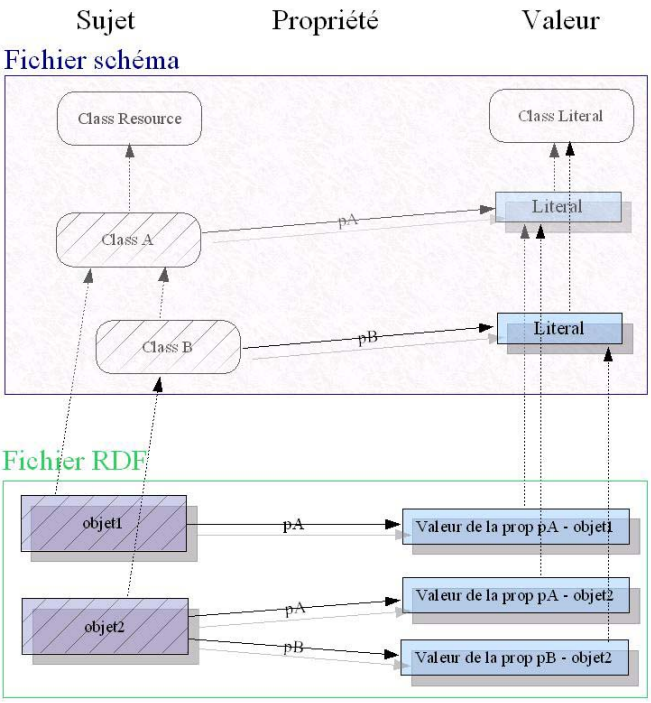
Les quatres déclarations précédentes pour être facilement réutilisables seront mises dans un seul fichier schéma RDF nommé *vocabulaire.rdf*.

Après la création du vocabulaire, nous pouvons maintenant créer des instances de nos classes. Nous allons créer un nouveau fichier RDF (fichier_ex.rdf) qui va importer le vocabulaire créé précédemment (le fichier *vocabulaire.rdf*), nous aurons deux objets :

- objet1 de Class A ayant comme valeur de la propriété *pA* la chaîne de caractères : "Valeur de la prop pA - objet1"
- objet2 de Class B ayant comme valeur de la propriété *pA* la chaîne : "Valeur de la prop pA - objet2" et "Valeur de la prop pB - objet2" pour la propriété *pB*

```
<?xml version="1.0" encoding="iso-8859-1"?>

<rdf:RDF
xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:vocabulaire="file:///vocabulaire.rdf#"
xml:base         ="file:///fichier_ex.rdf#"
>

 <rdf:Description rdf:ID="objet1">
  <rdf:type rdf:resource="file:///vocabulaire.rdf#A"/>
  <vocabulaire:pA>Valeur de la prop pA - objet1</vocabulaire:pA>
 </rdf:Description>

 <rdf:Description rdf:ID="objet2">
  <rdf:type rdf:resource="file:///vocabulaire.rdf#B"/>
  <vocabulaire:pA>Valeur de la prop pA - objet2</vocabulaire:pA>
  <vocabulaire:pB>Valeur de la prop pB - objet2</vocabulaire:pB>
 </rdf:Description>

</rdf:RDF>
```

Figure 5.6. Le fichier instancié



5. XML Schema (XSD)

XML est un langage extensible et chaque développeur peut créer ses fichiers XML pour stocker et partager de l'information. Utilisant cette facilité, les développeurs ont créé des documents où se trouvent une grande gamme d'informations. XML peut résoudre différents problèmes de partage d'information. Les points majeurs de ce processus sont la déclaration formelle et la documentation de ces formats; Ce qui fournit une base sur laquelle les développeurs peuvent ériger des applications.

XML Schema est un langage de formalisation des contraintes, il explicite la structure d'un fichier XML. Sa principale raison d'être est de décrire une structure précise et permettre la validation d'un fichier XML par rapport à cette description.

Voici un exemple de document XML :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="file:///document.xsd">
  <p id="1">Bonjour à tous et bienvenue sur le forum.</p>
  <p id="2">Je suis Jean Robert</p>
</document>
```

et son fichier schéma :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:element name="document">
   <xsd:sequence>
    <xsd:element name="p" minOccurs="1" maxOccurs="unbounded" >
     <xsd:complexType>
      <xsd:simpleContent>
       <xsd:extension base="xsd:string">
         <xsd:attribute name="id" type="xsd:integer" use="required"/>
       </xsd:extension>
      </xsd:simpleContent>
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
 </xsd:element>

</xsd:schema>
```

Le fichier schéma nous signifie ici que l'élément *document* est une séquence d'élement *p* (au moins 1). L'élément *p*, quand à lui, est de type complexe (il peut avoir des attributs), il possède un attribut *id* de type xsd:string (un entier) et son contenu (celui entre la balise et) est de type xsd:string (une chaîne de caractêre).

Plusieurs types simples sont définis de base dans XML Schema : xsd:byte, xsd:long, xsd:boolean,... et il évidemment possible de créer ses propres types. Par exemple le type *myxs:email* qui spécifierait une chaîne de caractères qui contiennent obligatoirement une "@".

En ce qui concerne la validation, plusieurs outils sont disponibles et peuvent permettre de vérifier cette conformité par rapport au schéma (XMLSpy, XXE, ...), nous entrerons plus en détail dans la **validation** dans le chapitre consacré au Outils utilisés.

6. Quel schéma utiliser?

Comme nous l'avons vu précédemment, XML est **LA technologie** qui nous permet de stocker et partager les données structurées. RDF est **LA technologie** de description et peut exploiter cette syntaxe XML. Pour exprimer les contraintes et différentes règles, XML dispose de son langage de schéma plus élaboré : XML Schema; et RDF : RDF Schéma.

Voici un tableau récapitulant ces deux langages de schéma :

Tableau 5.2. XSD et RDFS

	XML Schema (XSD)	**RDF Schema**
Modèle de données	Arbre XML	triplet RDF (resource, property, statement)
Stockage	Fichier balisé XML	Fichier N3, Table SQL (3-uplets), Ficher XML, ...
Notions Objets : Classes, ...	-	Concept de Classe et de Propriétés
Héritage	-	Sous-classe
Domaine d'un élément	-	Oui
Cardinalité d'un élément	Minimum et Maximum	-
Types de bases	entier, chaînes, ... des types peuvent être définis avec *XML Schema*	Le type *Literal* (simple chaîne de caractères)
Enumération de valeurs	Oui (Enumeration)	-
Ensemble d'élements	Oui (Union)	Oui (Bag ou Seq)
Outils de requêtage	XQuery et XPath	XQuery et XPath quand le fichier est RDF/XML. Dans les autres cas : RD-QL, SquishQL, ...
Librairies Java	De nombreuses implémentations de DOM et de SAX	RDF Suite et Jena RDF

XSD et RDFS offrent chacun leurs avantages; RDFS est plus adapté pour une description qui utiliserait un vocabulaire (une ontologie). Pour le contrôle des types, XSD est plus adapté. Mais il faut garder à l'esprit que RDF peut prendre la forme d'un fichier XML (le format RDF/XML), il pourra exploiter les DTD et autres schémas XML. La **composition** de ces **technologies** nous permettra donc de bénéficier d'un formalisme fort. Nous verrons par la suite que malheureusement peu d'applications de validation de schémas utilisent cette composition.

7. DocBook

DocBook comme le HTML sont des langages qui découlent du SGML. En d'autres termes si HTML était une valise, SGML serait une fabrique de valises. Le rôle de DocBook est de créer facilement des documents qui soient indépendants du matériel avec lesquels on les visualise. Ainsi, en DocBook, on précise le contenu sémantique du texte (paragraphes, parties, idées à souligner, etc) et non son apparence.

Oasis et *O'Reilly* (le célèbre éditeur) sont à la source de *DocBook*, celle-ci est une DTD particulière, c'est-à-dire la description des balises, leur syntaxe et leur enchevêtrement:

Ex :<title> ce met après une section, mais pas après un simple paragraphe.

DocBook est un standard de la **documention technique**. Son principal avantage vient du fait qu'il puisse être écrit en *XML* et de ce fait, à l'aide d'outils de transformation de type XSL et de feuilles de style spéciales, il est possible à partir d'un document *docbook* de produire un site web (avec table des matières, ...), un document Tex, un document RTF ou encore du PDF;

DocBook s'impose réellement comme un standard et nombreux sont les développeurs qui l'utilisent. Sa simplicité et le large panel de *formats de sorties* sont ses principaux atouts. Même si un document *docbook* est éditable à partir d'un simple éditeur de texte, il est préférable d'utiliser un outil spécialisé. Pour ma part, j'ai utilisé XXE de XMLMind (cf XXE) pour la rédaction de ce *rapport* et celle de la documentation technique du *prototype*.

Note

La DTD docbook est disponible sur http://www.docbook.org/xml/ et les feuilles de styles XSL sur http://docbook.sourceforge.net/projects/xsl/.

Chapitre 6. Outils utilisés

Pour mener à bien ce projet, nous avons eu recours à de nombreux outils que nous vous exposons dans cette partie.

1. Java de Sun et l'EDI11 JBuilder

1.1. Présentation

La Plateforme Java est une Plateforme très robuste qui offre l'avantage d'être portable sans re-compilation. Ainsi, une application Java fonctionnera de la même manière sur une plateforme WindowsNT et une plateforme type Unix ou Linux.

Coté "affichage" Java offre deux principales interfaces graphiques : Awt et Swing, ces deux interfaces "agréables" et "rapides" permettent une conception relativement aisée d'interfaces avancées.

Une multitude de bibliothèques sont disponibles en Java, des bibliothèques très bien documentées (JavaDoc) et d'une grande facilité d'utilisation.

L'environnement de dévelopement utilisé pour l'implémentation du prototype a été *JBuilder X Foundation*. Il s'agit d'une version gratuite du célèbre outil de Borland, celui-ci dispose d'une très belle interface (affichant les erreurs sans que nous ayons à compiler le fichier source), de très bon outils de Débuggage et d'un mode *Design* qui permet de manipuler facilement les composants graphiques.

1.2. Swing et Widgets de base

La bibliothèque *Swing* inclus tout les widgets (objets graphiques) de base nécessaires à la conception d'interfaces graphiques :

- Des boutons (JButton)
- Des titres (JLabel)
- Des zones de saisies/affichages de textes : zone mono-ligne (JTextField), multi-lignes (JTextArea) et même HTML (JEditorPane)
- Des boîtes permettant la sélection d'éléments parmi une liste (JComboBox)
- Des cases à cocher, ...

11EDI : Environnement de Développement Intégré

Deux autres composants ont également été utilisés pour l'interface du prototype : le composant *JTable* qui permet la gestion d'un tableau de deux dimensions et le composant *JTree* qui gère les objets arborescents. L'utilisation de ces widgets nécessite l'utilisation d'un *modèle Java* : il se charge de l'affichage du widget et permet d'interagir avec celui-ci.

1.3. Un nouveau widget

Même si Java nous propose dans ses JFC[12] un grand nombre de classes intérressantes, il nous est nécessaire, quand nous avons des besoins spécifiques, de créer nos propres composants.

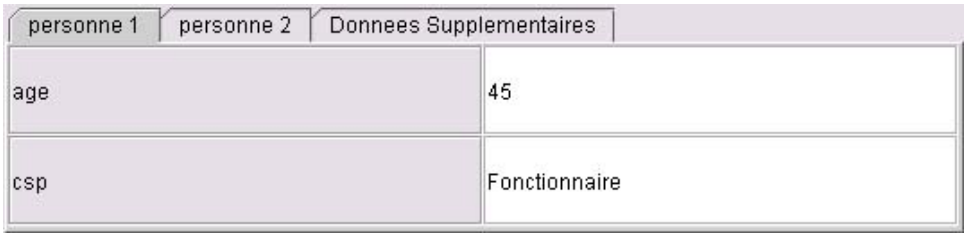
Voici l'exemple d'un composant utilisé lors du stage. Celui-ci permet à partir d'un document XML de créer un composant spécifique.

Fonctionnement du widget : le composant reçoit un document XML et crée en fonction des noeuds des tableaux et autres zones de texte.

```
<TEXMLAttList>

 <couples name='personne 1'>
  <item att='age' val='45'>
  <item att='csp' val='Fonctionnaire'>
 </couples>

 <couples name='personne 2'>
  <item att='csp' val='Fonctionnaire'>
  <item att='age' val='-1'>
 </couples>

 <couples name='Donnees Supplementaires'>
  <item att='personnes_au_foyer' val='4'>
  <item att='enfants_de_moins_de_15_ans' val='2'>
 </couples>

</TEXMLAttList>
```

Pour ce document il va créer un onglet pour chaque *couples*; on a donc 3 onglets ("personne 1", "personne 2" et "Donnees Supplementaires") et dans chaque onglet nous aurons les labels et zones de texte correspondant aux attributs contenus dans chaque *item*.

Figure 6.1. Le nouveau Widget



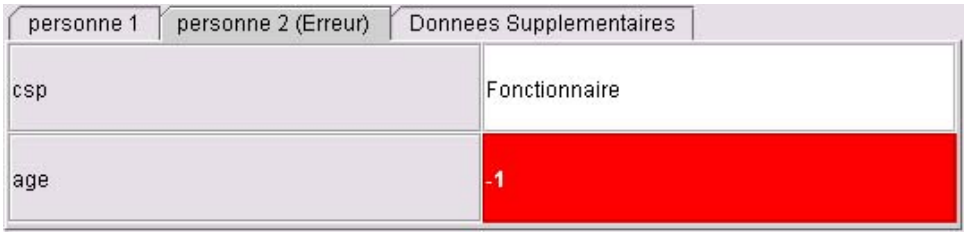
Le composant prend en entrée un document XML, le "parse" à l'aide de *DOM* (cf DOM) et renvoye un JPanel. Ce composant a été utilisé dans l'affichage des bases

12JFC : Java Foundation Classes

Dixit : on pouvait dans le cas d'une erreur mettre un attribut en surbrillance, il suffisait dans le document XML de rajouter un attribut "err".

```
<couples name='personne 2' err>
 <item att='csp' val='Fonctionnaire'>
 <item att='age' val='-1' err>
</couples>
```

Figure 6.2. Mise en surbrillance de l'erreur



2. La bibliothèque ORO de Jakarta

Après l'étude des données EDF, un premier prototype a été rapidement implémenté pour la lecture d'un forum. Le prototype utilisait le langage de script Perl et lisait un fichier ASCII.

Les expressions régulières Perl permettaient de distinguer dans le fichier forum les différentes sortes de données lues (nom de l'auteur, e-mail, message, ...)

Figure 6.3. Un premier prototype en Perl

```
if ($line =~ /^\$Revisions:  (.*)/){
   #28/11/2002 19:14:49,29/11/2002 09:59:56
   @date_and_time = split(',',$1);
   $date_and_time[0] =~ /(\d\d)\?(\d\d)\?(\d\d\d\d) (\d\d)\:(\d\d)\:(\d\d)/
   #                     $1      $2      $3          $4      $5      $6
   $date = "$3"."-"."$2"."-"."$1";
   $time = "$4".":"."$5".":"."$6";
}

if ($line =~ /^publier:  Oui/){
   $publish = 1;
}

if ($line =~ /^NomAuteur:  (.*)/){
   $author_name = $1;
}

if ($line =~ /^email:  (.*)/){
   $author_email = $1;
}

if ($line =~ /^Subject:  (.*)/){
   $subject = $1;
}

if ($line =~ /^body:  (.*)/){
   $body = $1;
   $body_mode = 1;
}

if ($line =~ /^□/){

   #on a toutes les données
   print outputl "\<message id=\"$id\" parent_id=\"$parent_id\"\>\n";
```

Durant l'étude des différentes sources de données, j'ai élaboré des tables comprenant les expressions *Perl* ainsi que les règles permettant l'extraction d'informations de ces sources. Lors de la conception de la plate-forme de structuration, j'ai utilisé la bibliothèque ORO qui offrait le support intégral des expressions régulières *Perl*, l'implémentation a ainsi été accélérée et une future modification du code (mise à jour ou débug) serait simplifiée.

Nous allons voir un exemple d'utilisation de la librairie pour le *match* et la *récupération de valeurs* :

Nous lisons la variable de type chaîne de caractères de nom "ligne".

Tableau 6.1. De Perl à ORO

	Perl	ORO
déclaration	$ligne = "NomAuteur : Jean Claude"; nom_auteur = "";	String ligne = "NomAuteur : Jean Claude"; String nom_auteur =""; Perl5Util reg_expr = new Perl5Util();
match (si la ligne commence par "NomAuteur :")	if($ligne =~ /^NomAuteur: (.*)/)	if (reg_expr.match("/^NomAuteur: (.*)/", ligne)
récupération du nom de l'auteur, il s'agit de "(.*)" de l'expression	$nom_auteur = $1;	nom_auteur = reg_expr.group(1);

3. DOM de W3C

DOM (Document Object Model) est une spécification du W3C définissant la structure d'un document sous forme d'une hiérarchie d'objets, afin de simplifier l'accès aux éléments constitutifs du document. Plus exactement, DOM est un langage normalisé d'interface (API[13]), indépendant de toute plateforme et de tout langage, permettant à une application de parcourir la structure du document et d'agir dynamiquement sur celui-ci. Ainsi nous pouvons utiliser DOM pour naviguer au sein d'un document XML, ce qui nous permettra par exemple de pouvoir récupérer le contenu d'un noeud, le modifier.

Il existe un autre type de *parser* XML : *SAX*; la différence entre cette technologie et *DOM* vient du fait que *DOM* charge l'intégralité du document en mémoire avant de travailler dessus alors que *SAX* lance des actions (trigger) au fur et a mesure qu'il lit le document XML. Quand le fichier est volumineux, l'utilisation de *SAX* est préférable à celle de *DOM*, mais quand il est nécessaire de naviguer dans la structure XML, *DOM* est plus beaucoup plus adapté.

13API : Application Programming Interface

Note

Il existe plusieurs niveaux de spécifications DOM ; le site du W3C consacré à DOM (http://www.w3.org/DOM/) nous présente celles-ci.

4. Jena RDF Api des Laboratoires Hewlett-Packard

Jena est une bibliothèque qui regroupe des méthodes permettant l'utilisation des fichiers RDF dans une application Java. Il fonctionne de la même manière que *DOM* : le modèle chargé en mémoire.

Cet API permet de :

- Manipuler du RDF
- Charger des modèles RDF en mémoire, y accéder et le modifier
- Requêter un modèle à partir de méthodes conçues à cet effet ou à l'aide du langage RD-QL (plus de détails dans la partie suivante)

Cette API défini un objet de type *Model* qui est la représentation mémoire d'un fichier RDF, c'est-à-dire la représentation du graphe constitué d'un ensemble de ressources et d'assertions qui s'y rattachent. *Jena* charge le modèle depuis un fichier et nous pouvons ensuite le manipuler.

Suivons un exemple de modèle :

```
//Declaration de l'objet
Model model;

//creation d'un modèle vide en mémoire :
model = ModelFactory.createDefaultModel();

//chargement du modèle a partir d'un fichier
model.read(new FileInputStream("analyse.rdf"),"");

//navigation dans le modèle : listage des ressources
ResIterator iterator_ressources = model.listSubjects();

while(iterator_ressources.hasNext()){
  Resource resource = iterator_ressources.nextResource();
  //affichage de la resource
  System.out.println(resource.toString());
}
```

Il est possible à partir d'un modèle de créer des resources et lui ajouter des propriétés :

```
//creation d'une resource typée comme une enquete EDF
Resource ma_resource = model.createResource(EDF.Enquete);
```

```
//ajout d'une propriété
ma_resource.addProperty(EDF.mene_par,"Jean Vidal");

//sauvegarde du modèle
model.write(new FileOutputStream("analyse.rdf");
```

Enfin *Jena* permet l'utilisation de vocabulaire qu'on a soi-même définis, *Jena* propose pour cela un outil très pratique appelé `schemagen`, qui va prendre en entrée la description d'un vocabulaire (un fichier RDFSchema) et produire la classe Java correspondante.

5. RD-QL

RD-QL ou RDF Data Query Language est un langage de requêtes RDF. Celui-ci permet d'extraire des données d'une base RDF en utilisant un langage similaire à celui des bases de données relationnelles : le *SQL*.

L'utilisation du RD-QL est simple, il suffit de garder en mémoire que nous travaillons sur des *triplets*. Voici l'exemple d'une requête sur un "forum" qui nous permet de récupérer les messages (leurs *URI*, Ex : message0001.xml) écrits par "Jean Robert".

```
SELECT ?resource
FROM file://forums/ex_forum.rdf
WHERE (?resource, forum::auteur, ?personne)
      (?personne, personne::nom, "Jean Robert")
USING rdf      FOR http://www.w3.org/1999/02/22-rdf-syntax-ns#
      forum    FOR file://schema_p00f/forum.rdf#
      personne FOR file://schema_p00f/personne.rdf#
```

Nous remarquons que la clause *WHERE* implique l'utilisation de triplets et qu'il est possible d'utiliser des "espaces de nom" ("*espace_de_nom*::*propriété*") en les définissant dans la clause *USING*.

Note

Jena intègre un moteur RD-QL, il est accessible en ligne de commande par l'intermédiaire de la classe **jena.rdfquery**.

6. Le validateur VRP de l'ICS-Forth

Comme nous l'avons vu dans le chapitre précédent, la validation d'un document est une phase importante de ce projet. De la même manière que les fichiers XML, il est nécessaire de valider les fichiers RDF. Deux outils ont été comparés : RDF Validator et VRP de ICS-Forth.

1. RDF Validator (ARP2)

   Ce validateur RDF (http://www.w3.org/RDF/Validator/) est le validateur fournit par le W3C, il utilise les bibliothèques ARP2 inclus dans Jena. La **validation** d'un document RDF par ce validateur est assez **sommaire**. En effet, ARP2 n'effectue qu'une validation syntaxique : il s'assure que le document est correctement structuré (balises xml) et vérifie si le vocabulaire utilisé après les balises *rdf* et *rdfs* respecte les normes RDF; dans le cas de

l'utilsation de vocabulaires externes, ARP2 n'effectue pas de contrôle.

Le prototype repose sur des modèles que nous avons définis, ARP2 ne satisfait donc pas nos besoins en terme de validation sémantique; de plus, le "contrôle de type" nécessaire pour garantir la bonne transformation d'une source n'est pas présent dans ce validateur. Il nous est alors impossible d'assurer que le fichier RDF que nous avons créé est réellement correct.

2. VRP de ICS-Forth (University of Crète)

   VRP (http://139.91.183.30:9090/RDF/VRP) est le validateur que nous avons utilisé pour l'implémentation de l'application. Celui-ci ne se contente pas de contrôler la structure du fichier RDF (validation syntaxique) à l'instar d'ARP2, il importe les fichiers *vocabulaires* et effectue des contrôles sémantiques pour les classes et leurs propriétés. Il intègre également un validateur de type XSD qui permet de valider les documents RDF à la manière des fichiers XML : nous pouvons utiliser les types simples définis par les schémas W3C dans nos fichiers RDFS.

   En plus d'offrir la validation syntaxique, sémantique et le support des types XSD, VRP génère un fichier rapport facilement réutilisable. C'est effectivement ce fichier qui nous permet de repérer les erreurs contenues dans les forums et autres base "dixit".

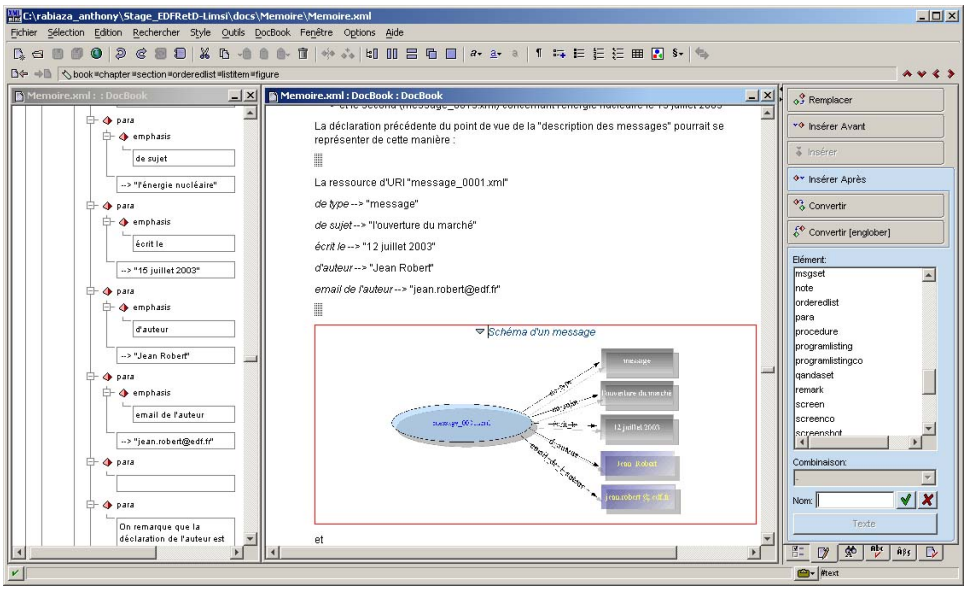
7. XXE de XMLMind

XXE ou XMLMind XML Editor est un très bon outil d'édition XML; de la même manière que les meilleurs boîtes à outils de ces documents balisés (XMLSpy d'Altova, XMLBuddy ou Emacs), celui-ci édite les fichiers de type XML en WYSIWYG[14] , complète la saisie des balises et les valide selon une DTD ou un autre type de schéma. Ses principaux avantages sont sa portabilité (fonctionne sous Java) et sa gestion des fichiers *docbook*; il permet alors d'écrire des documents conformes à la *DTD docbook* comme avec votre traitement de texte favori.

L'interface de ce logiciel comporte une vue arborescente ainsi qu'une vue WYSIWYG du document; l'utilisation est intuitive et nous avons facilement accès à toutes les fonctionnalités disponibles : vérification de l'orthographe, validation et conversion. Ainsi, il est possible d'exporter le fichier docbook en RTF, PDF et même HTML; un *parser* est intégré à XXE et permet la transformation par l'intermédiaire de feuilles de style XSL.

14WYSIWYG : What You See Is What You Get

Figure 6.4. XXE



Note

Il est cependant regrettable que dans la version standard (gratuite), seule la conversion en HTML est disponible. Pour remédier à cela, il suffira de télécharger les feuilles de style docbook (http://docbook.sourceforge.net/projects/xsl) et le formateur FOP (http://xml.apache.org/fop/).

Chapitre 7. Architecture du prototype

Le prototype qui a été implémenté répond lettre par lettre au cahier des charges. Celui-ci devait, grosso modo, structurer les différentes sources et permettre la validation et modification les fichiers obtenus. Le prototype implémenté en Java est constitué de plusieurs modules :

1. Modules fonctionnels

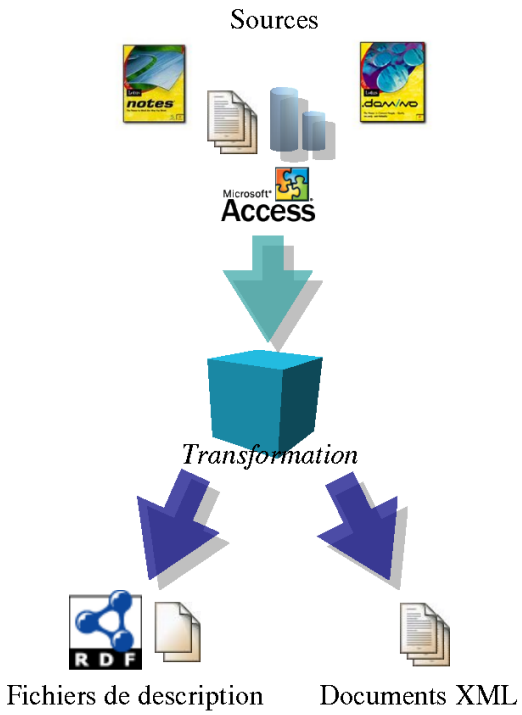
Nous allons à présent voir les différents modules qui composent l'application. Ils reposent sur les librairies Java de bases (java.io et java.util), la librairie ORO et les classes JenaRDF.

On peut diviser l'application en deux familles de modules :

- La brique de transformation (ou structuration) qui se charge de lire les différentes sources et de générer les fichiers de descriptions RDF et documents XML.
- Les briques VVM (Visualisation, Validation et Modification) qui permettent de réexploiter les fichiers RDF générés.

1.1. La brique de transformation

Figure 7.1. La brique de transformation



Cette brique :

1. prend en entrée des sources hétérogènes.

   Lecture d'un flux de caractères par l'intermédiaire de la classe Java *java.io.DataInputStream*. (Remarque : ce flux peut provenir d'un réseau)

2. extrait les méta-informations contenues dans le flux

   Utilisation des expressions régulières Perl de ORO de la classe *org.apache.oro.text.perl.Perl5Util*.

3. structure ces informations et génère les fichiers de description

   La génération s'effectue par l'intermédiaire de la classe *com.hp.hpl.jena.rdf.** et les modèles associés.

4. structure les documents de type texte et génère les documents XML.

   La classe *com.EdfLimsi.p00f.structuration.DocumentTransformer* prends en entrée des fichiers textes et produit des documents XML avec paragraphes.

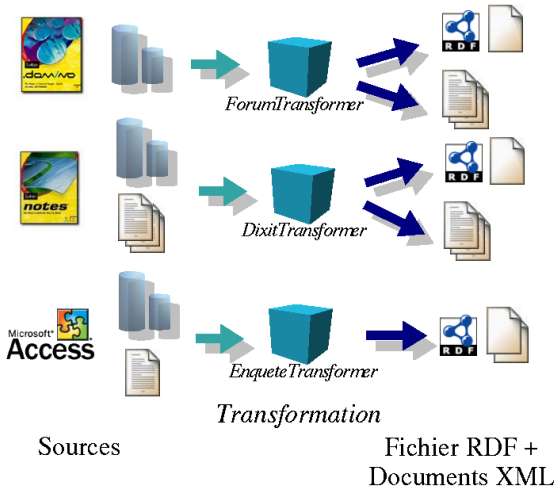
Nous vous rappelons les sources :

- Un forum de discussion extrait d'une base *Domino*.
- Une base de données d'entretiens capitalisés provenant d'une base *Notes*: une fichier export ainsi que les guides d'entretien et entretiens au format *texte*.
- Une enquête téléphonique : un questionnaire au format *texte* et une table *Access.*

Pour chacune des sources est générée :

- Un (1) fichier de description RDF et autant de documents XML que de messages dans le forum.
- Un (1) fichier de description RDF ainsi que des documents XML provenant de la conversion des guides d'entretiens et entretiens.
- Un (1) fichier de description RDF décrivant le questionnaire et x fichiers RDF contenant les données de l'enquête. avec x = *1 + partie_entière(nombre d'interviews / 1000)*.

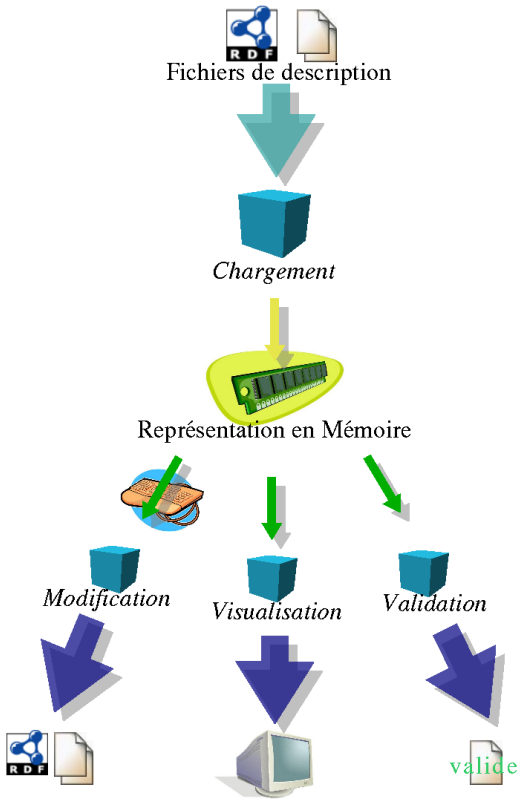
Figure 7.2. Les différents modules de la brique Transformation



1.2. Les briques VVM

Afin de manipuler les fichiers générés par cette première brique, nous utiliserons les briques VVM (Visualisation, Validation et Modification). En effet, malgré le fait que les fichiers de description RDF soient au format XML; format permettant la lecture et la modification du document par l'utilisateur en utilisant un simple éditeur de texte; il est assez difficile de modifier les fichiers RDF/XML "à la main", il a donc été nécessaire d'implémenter des outils manipulant ces fichiers.

Figure 7.3. Les briques VVM (Visualisation, Validation et Modification)



Avant d'utiliser les briques VVM, il est nécessaire de "charger" le fichier RDF : l'outil *JenaRDF API* nous permet de lire les fichiers RDF et d'en avoir une représentation en mémoire. Une fois le modèle "chargé", il est possible de l'afficher ou de le modifier.

- La brique *Visualisation* reprend la représentation en mémoire et crée, selon le type du document, une représentation graphique adaptée au type de la source (arbres, tableaux, etc.)
- La brique *Validation* "contrôle" le fichier RDF : dans le cas ou il trouverait des erreurs, il informe l'utilisateur de la présence et du type de celles-ci et met en surbrillance les données incorrectes.
- La brique *Modification* intervient avec la brique Visualisation : l'utilisateur peut modifier la représentation en mémoire du fichier par l'intermédiaire de l'interface de visualisation; la brique Modification permet alors de prendre compte de ces changements : il effectue la mise à jour du fichier RDF.

1.2.1. La brique Visualisation

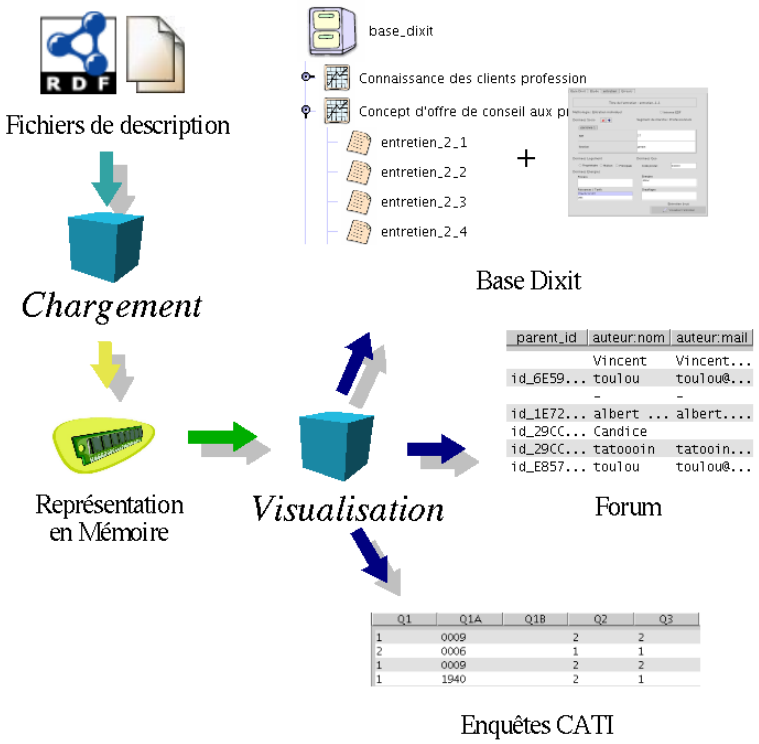
La brique de visualisation est composée de 3 sous-briques et chaque sous-brique est associée à un modèle Java spécifique.

La première sous-brique : *TEForumNavigator* permet la "navigation" dans un forum, elle inclut le widget Swing *JTable* pour l'affichage des données sur le forum et un composant *JTextArea* pour l'affichage des messages XML.

La seconde : *TEDixitNavigator* est utilisée pour une base Dixit, le widget Swing *JTree* hiérarchise les classes contenues dans le RDF et un widget personnalisé sert pour l'affichage des études et celle des entretiens.

Finalement, le composant *TEEnqueteNavigator* interface une enquête CATI. Ce dernier intègre le widget *JTable* comme dans *TEForumNavigator* mais il s'agit ici d'une version plus élaborée qui prend en compte l'affichage des *ToolTips*15 selon la position de la souris.

Figure 7.4. Les méthodes d'affichage selon le type du fichier de description lu



15ToolTips : info bulle

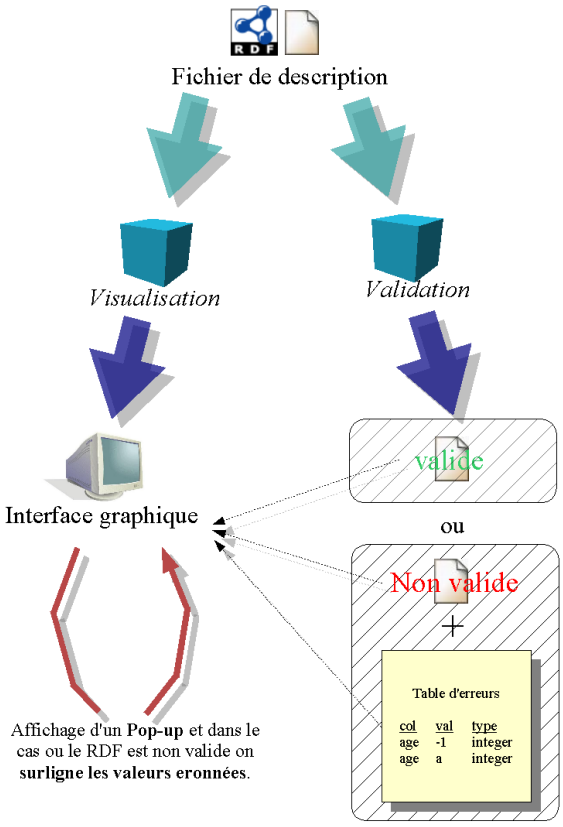
1.2.2. La brique Validation

Cette brique repose sur la classe *gr.forth.ics.vrp.corevrp.Main* de la librairie VRP d'ICS-Forth et sur un parser *DOM* de W3C. En effet, l'outil d'ICS (cf VRP) permet de détecter les erreurs (sémantiques, syntaxiques, etc.) contenues dans un fichier RDF. On récupère le résultat de cette analyse et l'on génère une table qui va contenir les informations sur les erreurs.

Voici les différentes phases de la validation :

- post-validation : l'utilisateur charge un fichier RDF et clique sur le bouton **Valider**
- validation : chargement de la classe VRPValidator qui utilise l'outil d'ICS; celui-ci retourne le nombre d'erreurs et génère le tableau décrivant celles-ci
- affichage d'un pop-up informant l'utilisateur de la validité du fichier ou l'avertissant de la présence d'erreurs
- si nombre d'erreurs > 0 alors on surligne (d'une police large et rouge) les valeurs incorrectes que l'on a décelées

Figure 7.5. Validation et retour d'information sur l'interface

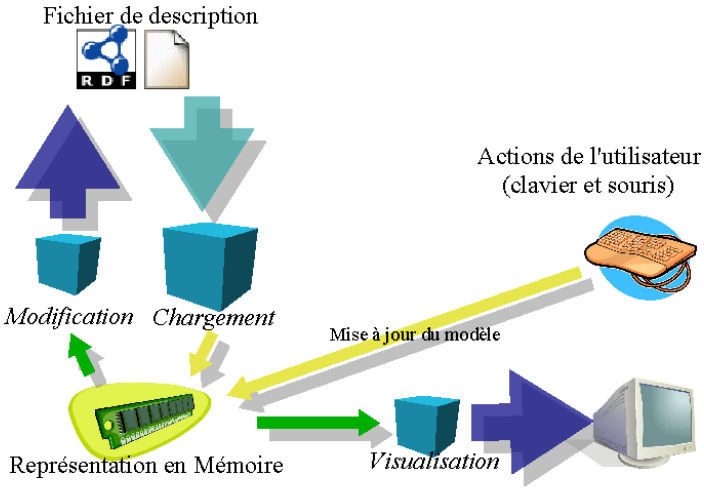


1.2.3. La brique Modification

De la même manière que le module de transformation, ce module reprend la représentation en mémoire des données et met à jour le fichier RDF.

L'utilisateur intervient sur cette représentation en utilisant les périphériques d'entrées (clavier et souris); ces actions sont instantanément répercutées à l'écran par la brique Visualisation; une fois que l'utilisateur a demandé un enregistrement, des procédures effectueront la conversion entre le modèle en mémoire et le format à écrire.

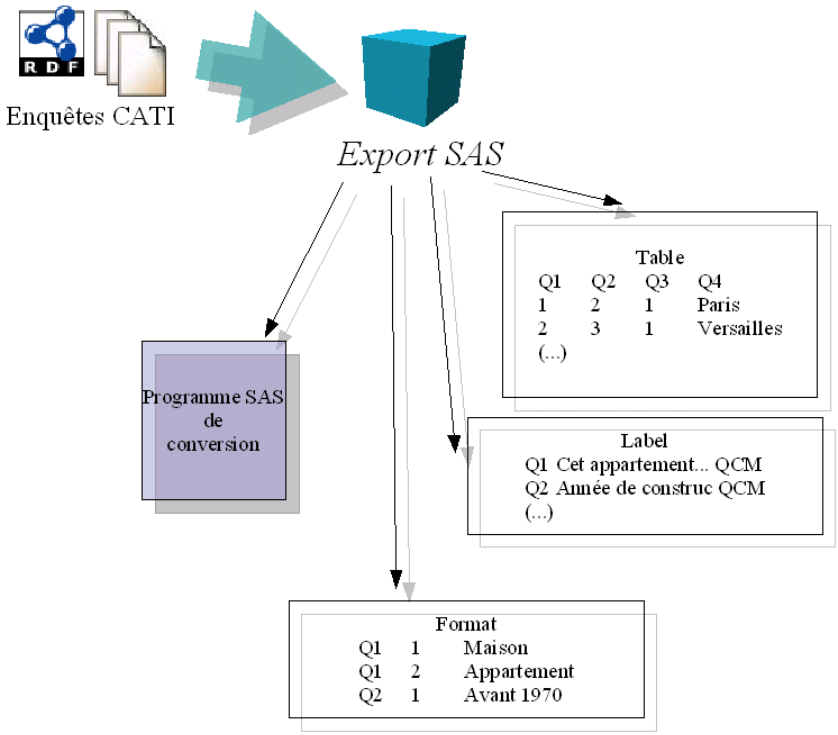
Figure 7.6. Modification des données et mise à jour du fichier



1.2.4. Export SAS

Le format RDF/XML permet un export facile dans n'importe quel format. Exporter les enquêtes CATI en format SAS n'a donc pas été une tâche difficile. Cet export a pour objectif de faciliter le travail des statisticiens qui travaillent principalement avec cet outil.

Figure 7.7. Le module d'exportation SAS



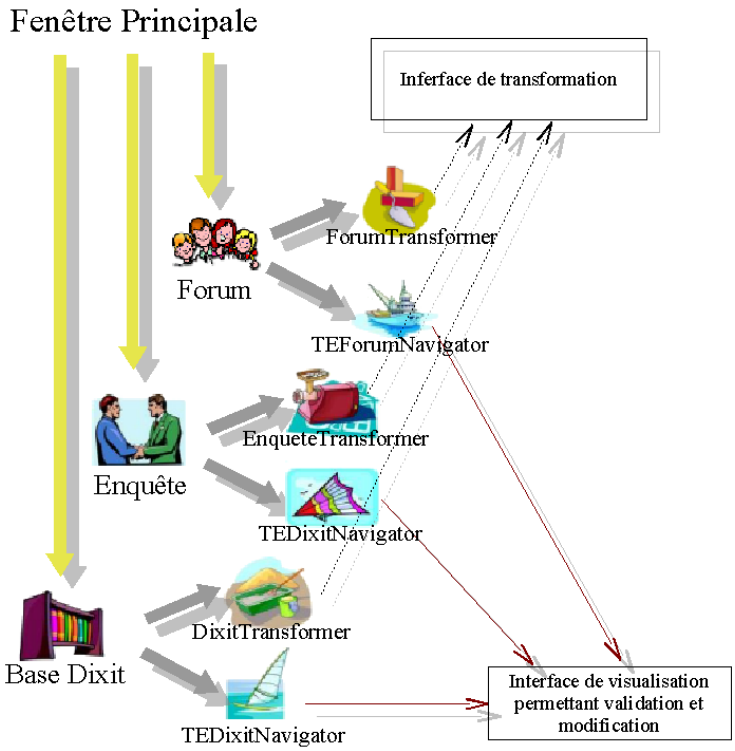
La module "Export SAS" produit une table brute, et différents fichiers (sur les titres et types, les significations) et un programme SAS. Ce dernier permet de mettre en forme la table à l'image de ce que fait l'interface de navigation CATI (cf Interface graphique) : nous avons alors la signification des divers codes de réponses, l'intitulé des questions, etc.

2. Interface graphique

Une interface graphique permet de manipuler facilement les briques précédentes. Dans cette partie, nous allons "naviguer" dans cette interface et décoder les mécanismes internes mis en place.

Figure 7.8. Fenêtre d'accueil

La fenêtre d'accueil nous donne accès à toutes les fonctionnalités de l'application. Voici un schéma hiérarchique qui nous présente cela :



Hiérarchie des composants

Le survol d'un des trois bouton "sources" entraîne l'affichage des options qui lui sont rattachées. Pour chaque source, nous avons les boutons *Transformer* et *Navigator* associés.

2.1. TransformationInterface

Ce composant, comme son nom l'indique, nous permet de "transformer" les différentes sources. Son interface associée est composée de différentes parties nous permettant de :

1. Définir les sources; selon le type des sources, nous avons des champs pour sélectionner :

   - un fichier brut (export domino) pour un forum de discussion
   - un fichier de données csv + un questionnaire pour une enquête CATI
   - un nom de répertoire (qui contient les guides et entretiens) + un fichier brut (export notes) pour une base dixit

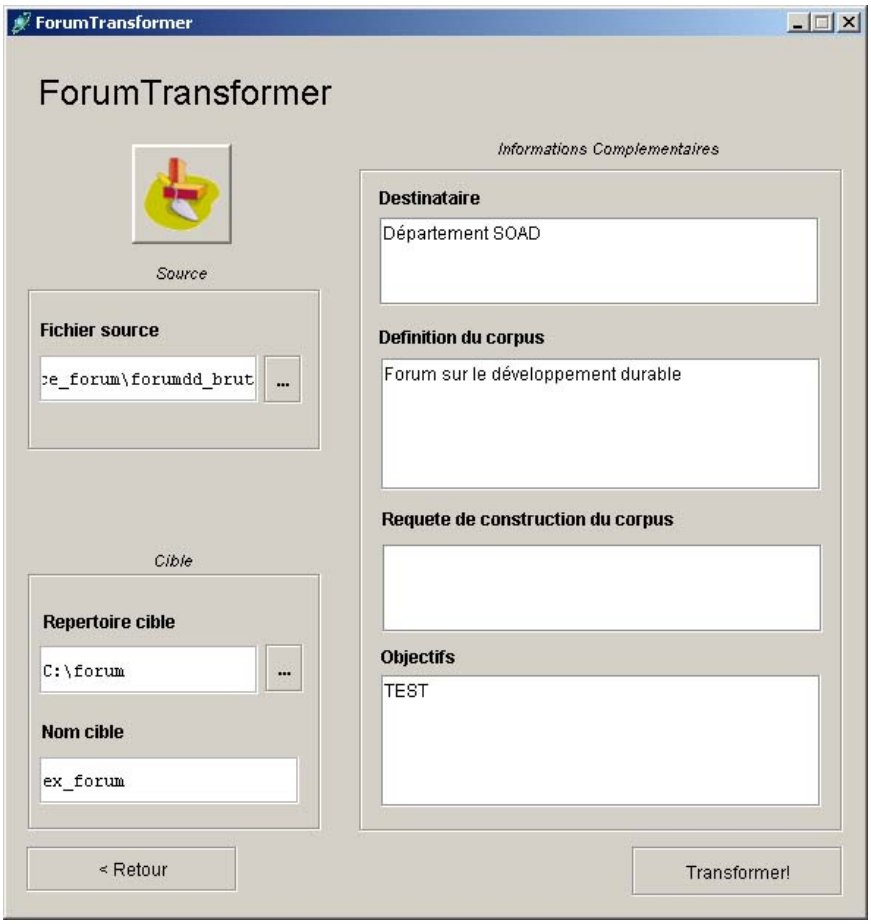
2. Sélectionner les cibles :

   - un répertoire de sortie
   - le nom du fichier RDF cible

3. Rajouter des méta-informations sur le fichier cible

   - le destinataire de l'étude
   - la définition du corpus
   - la requête de construction du corpus
   - les objectifs de l'étude

Figure 7.9. TransformationInferface pour un forum



Une fois les différents champs complétés, il suffit de cliquer sur **Transformer** pour lancer la procédure de transformation. Un "pop-up" nous informera du lancement de cette dernière et de son succès dans le cas où il se termine convenablement.

2.2. Le navigateur "Forum"

Ce composant permet la navigation dans un forum; il intervient après la sélection d'un fichier RDF par l'utilisateur. Nous pouvons trouver dans TEForumNavigator :

- une table contenant les méta-données sur une enquête : identifiant, identifiant du message père, nom de l'auteur, e-mail de l'auteur, sujet du message, ...

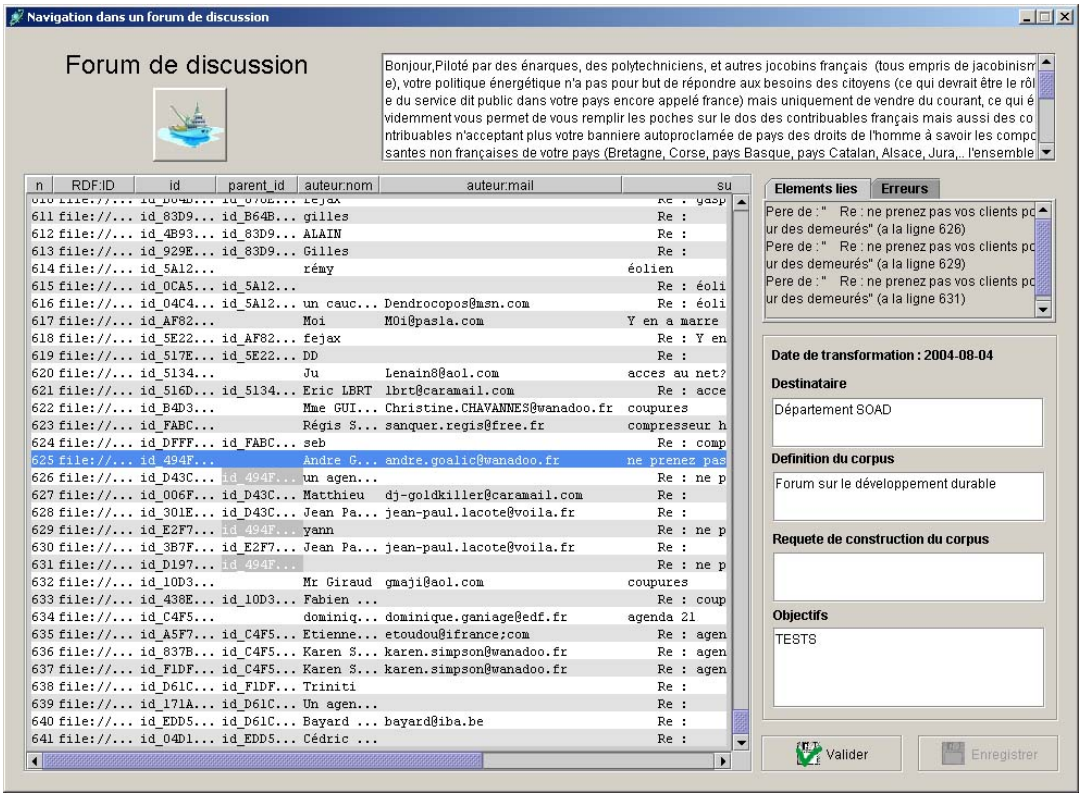
  Cette table permet : le déplacement ou changement de taille des colonnes, la sélection d'un message (pour l'affichage de son contenu), la recherche des messages fils (clic sur l'identifiant), la recherche du message père (clic sur l'identifiant du père) et la modification des méta-données.

  Le dernier champ de ce tableau nous informe de la modification des méta-informations d'un message par un opérateur : la case sera alors cochée et le nom de l'utilisateur affichée.

- une zone de texte qui affiche le contenu du message sélectionné (clic dans la table précédente)

- un panneau qui affiche les fils ou le père d'un message
- un panneau qui affiche les erreurs contenues dans le forum (après une validation)
- les méta-données sur le forum (saisies par l'utilisateur lors de la transformation)

Figure 7.10. Navigation dans un forum

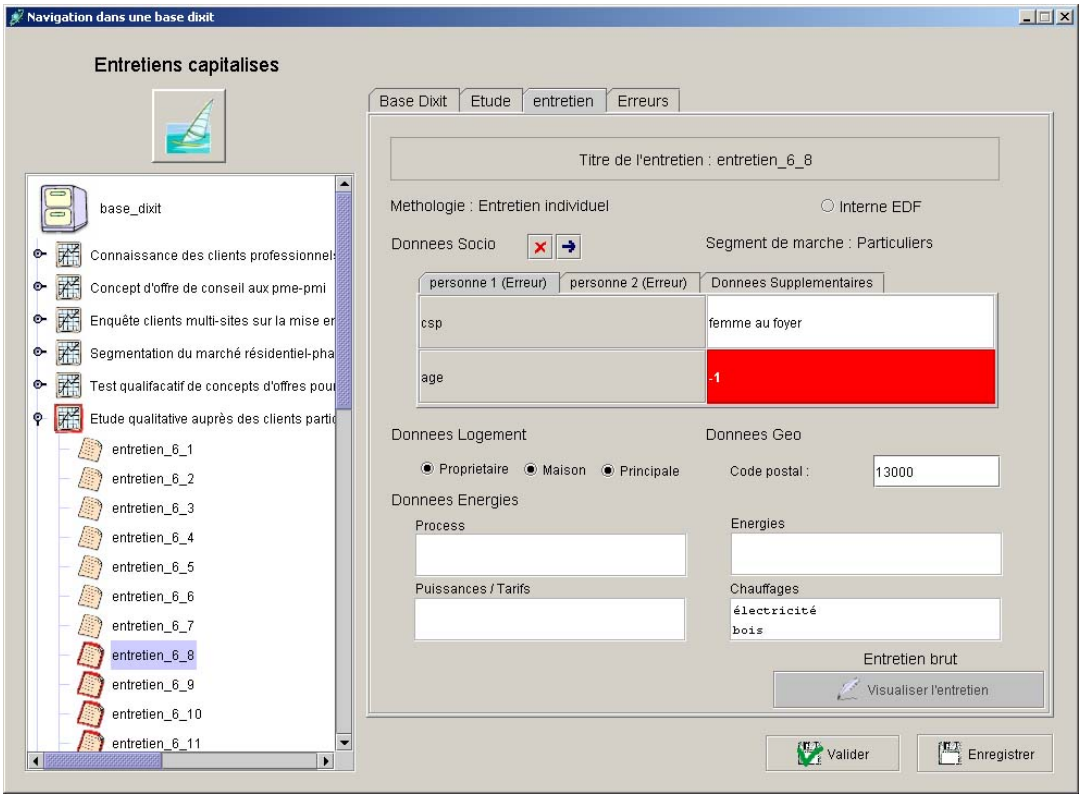


Cette interface permet également l'enregistrement des modifications et la validation du fichier RDF. Dans le dernier cas, les erreurs seront listées dans une zone conçue à cet effet et surlignées dans la table des méta-données.

2.3. Le navigateur "Dixit"

Ce navigateur nous permet de nous retrouver dans la structure complexe que comporte la base "Dixit". Un menu arborescent affiche un descriptif sommaire des études et des entretiens, et un autre composant permet d'avoir plus de détail sur ces éléments.

Figure 7.11. Navigation dans la base Dixit



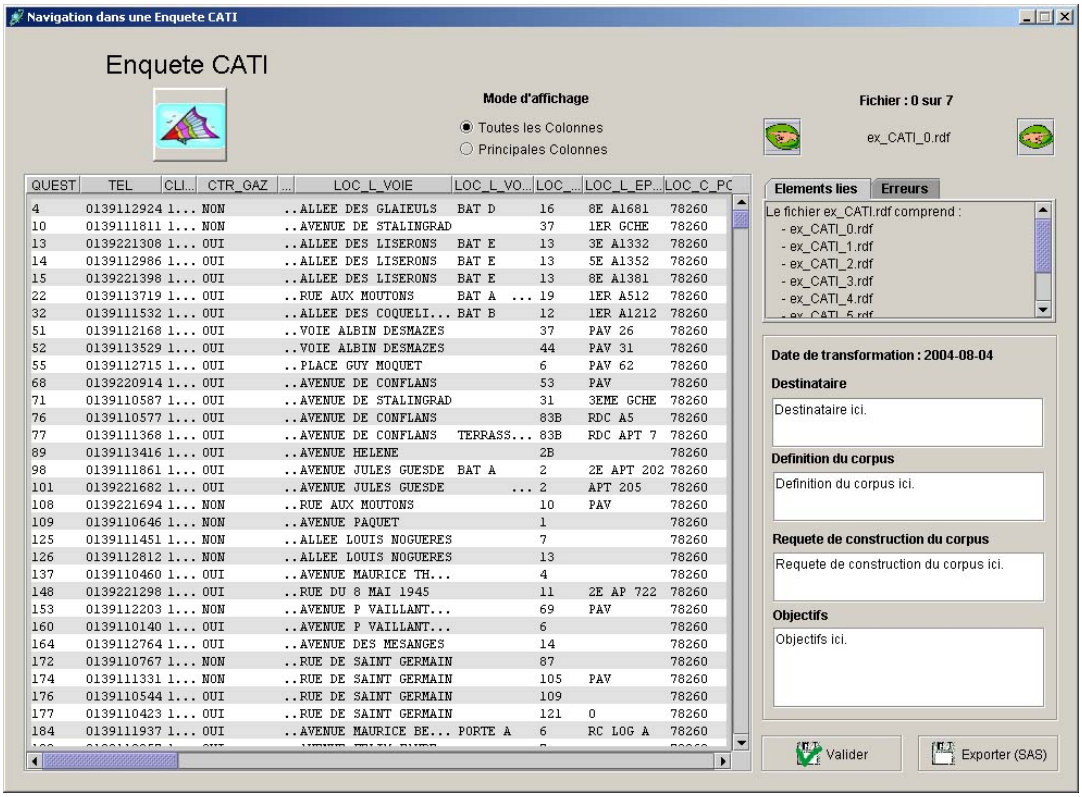
Cette interface permet également l'affichage des données rattachées à l'étude ou à l'entretien par l'intermédiaire d'un visualiseur XML. Celui-ci fonctionne de la même manière que le visualiseur de messages du navigateur "Forum".

Il est possible, comme dans le cas d'un "forum", de modifier des méta-données, de mettre à jour et de valider le fichier RDF. Durant la validation, l'interface joue un rôle majeur pour indiquer les éléments mis en cause. Par exemple si dans l'entretien *y* de l'étude *x*, la variable *âge* n'est pas de type entier, le noeud de "l'entretien *y*" ainsi que celui de son père (le noeud "étude *x*") seront surlignés en rouge. L'opérateur n'aura donc pas besoin de développer toutes les études pour retrouver l'entretien mis en cause.

2.4. Le navigateur "Enquête CATI"

Cette interface est similaire à celle utilisée pour l'affichage des forum, la différence réside dans le fait que les "statisticiens" pourront facilement "décoder" le résultats des enquêtes CATI. En effet, ce composant ne se contentera pas d'afficher les tableaux d'enquêtes, il expliquera à l'utilisateur les différentes significations des codes.

Figure 7.12. Navigation dans les enquêtes CATI



Exemple d'info-bulle

Les info-bulles sont affichées dans deux cas : soit quand la souris survole l'en-tête du tableau, on affiche alors l'intitulé de la question et l'éventuelle remarque, soit en survolant une case du tableau, la signification du code de la réponse est alors indiquée.

L'interface propose l'exportation des enquêtes en format SAS, ceci permettra d'avoir une interface similaire à la notre dans les outils SAS.

Chapitre 8. Perspectives

1. Contraintes et vocabulaire

Le prototype utilise des vocabulaires exploitant les types et contraintes RDF, RDF Schema, les types simples W3C (date, time, integer, ...) et deux Classes que nous avons créées : pack_metadonnées et personne. Toute source (forum, dixit, enquête) est une sous-classe de pack_metadonnées, cette classe possède des propriétés propres utilisables quelle que soit la source. La classe personne quand à elle, est réutilisée dans toutes les sources où intervient une personne (un interviewé dans "dixit", un internaute dans les "forum"), il est alors possible, sans connaître le type exact de la source, d'effectuer des requêtes sur l'atome personne.

Pour normaliser le nom des variables de sources différentes, il serait intéressant d'avoir recours au langage DAML+OIL, celui-ci permettrait les équivalences (daml:sameClassAs) ou encore une "énumération de possibilités" (oneOf) supérieure à celle utilisée avec RDFS.

2. Proposition d'architecture

La version actuelle du prototype fonctionnel est mono-poste, une version multipostes ne demanderait pas beaucoup de modifications. En effet, Jena intègre dans la version actuelle des méthodes permettant l'extraction des descripteurs depuis une base de données relationnelles (Oracle, MySQL, ...); les principales modifications à apporter seront plutôt au niveau du stockage de données (fichiers XML) car leurs stockage devraient également se faire sur une base de données. Les outils de visualisation et de requêtage devront alors extraire les fichiers XML de la base avant de pouvoir les utiliser.

Une autre solution consisterait à la mise en place d'un serveur Web par l'intermédiaire du serveur Joseki (http://www.joseki.org/). Ce dernier permet la publication de fichiers RDF sur la toile et ainsi que la connexion simultanée de plusieurs clients.

3. Propositions de formats

Pour faciliter la structuration des "enquêtes", il serait préférable d'utiliser une (1) structure référence. Celle-ci est utilisé dans certaines enquêtes "Dixit", nous avons :

- Q : texte , quand il s'agit de *l'interviewer*

- R : texte , quand il s'agit de *l'interviewé,* si plusieurs personnes sont interviewer alors on utilisera : **- R (nom_de_la_personne) : texte**

Les commentaires seront mis entre guillemets (Ex : "La personne se lève").

Il est nécessaire que le format du fichier de l'enquête puisse permettre une exportation "propre" en `txt` : il ne doit comprendre ni image ni symbole graphique (puces, ...).

Conclusion

J'ai été enchanté d'effectuer ce stage. Participer à ce projet **EDF R&D-Limsi** m'a permis d'évoluer dans le monde de l'entreprise et dans celui de la recherche; ces deux mondes se sont cottoyés tout au long du stage et m'ont permis de mieux appréhender le travail d'équipe et la collaboration entre professionnels et chercheurs : experts provenant d'univers différents et qui poursuivent un objectif commun.

En effectuant toutes les phases du projet, du recueil des besoins jusqu'à la mise à disposition de l'application, je me suis senti fortement impliqué par la qualité finale du produit et l'impact qu'il a eu auprès des différents départements. De plus, j'ai pu cerner l'importance qu'il fallait accorder au travail d'étude et de recherche effectué en amont du projet pour développer une application complète, extensible et répondant aux besoins du commanditaire.

Les technologies utilisées durant ce stage sont nouvelles et néanmoins remarquables. Plusieurs d'entre elles ne sont qu'en phase de test mais offrent pourtant des possibilités incomparables. Nous pouvons citer l'exemple du *RDF*, langage du Web sémantique, qui prend de l'ampleur et ne tardera sûrement pas être intégré aux moteurs de recherche actuels.

En ce qui concerne le bilan des réalisations, tous les objectifs fixés ont été atteints : les différents entrepôts de données ont été formalisés et validés, et des essais de "requêtage" sur les nouvelles bases ont été effectués. La plateforme alliant *Java* et *RDF/XML* nous offre vitesse, efficacité et robustesse en terme de structuration des sources et en terme de requêtage. Les différents commanditaires ont **testé** et **validé** l'application.

Finalement, je peux conclure par le fait que ce stage a été le complément parfait de mon DESS, tout le savoir-faire que j'ai acquis durant mon DESS m'a aidé à la réalisation de ce projet. Toutes les disciplines abordées ont été appliquées : les Technologies de l'information (Base de données, Datawarehouse, Extraction de connaissances à partir des BD), les outils de traitement de l'information (Indexation et Recherche d'Information), les méthodes de visualisation (Interface Homme-Machine, Visualisation de l'Information), les cours de communication et l'anglais (documentations techniques uniquement disponibles dans la langue de Shakespeare).

Annexe A. Questionnaire

1. Questionnaire

   a. Profil de l'utilisateur

   - Quelles sont les compétences informatiques mises en oeuvre dans l'utilisation des outils : minimales (utilisation du logiciel dans la configuration par défaut), intermédiaires (administration, paramétrages fins) ou expertes (programmation avec accès aux API).
   - Quel est son système d'exploitation : Unix, Linux, Mac ou Windows.
   - Quels sont les logiciels qu'il utilise : y a-t-il un outil plus fréquemment utilisé que les autres ? Quels sont les logiciels utilisés occasionnellement?
   - Quelles sont ces habitudes de collaboration : plusieurs sur un même projet ? relative autonomie ? Indiquer éventuellement la manière dont des personnes qui ont l'habitude de travailler ensemble se répartissent le travail.
   - Quelles sont ces connaissances sur le projet EDF/Limsi en cours ? Il lui est possible de formuler un besoin ou une attente par rapport à ce projet.
   - Qu'aimerait-il pouvoir faire ou qu'est-ce qu'il ne peut pas faire (ou faire difficilement) ?
   - Qu'est-ce que les gens accepteraient de changer dans leurs manière de travailler qui leur serait utile sans être trop coûteux? (limite de l'acceptable/ inacceptable)
   - Qui maintiendrait/administrerait éventuellement un nouvel outil ?

   b. Traitement des données

   - Comment s'effectue l'accès aux données? s'agit-t-il d'une base de données, de fichiers (local ou distant) ?
   - Quelle est la « durée de vie » des données ? (Ex. Les résultats d enquête ne sont plus exploités après x années). Il y a-t-il des traitements de séries temporelles ?
   - Est-ce que l'utilisateur d'une base ou d'un ensemble de données est aussi celui qui a constitué cet ensemble ? (il existe différents cas de figure)

- Comment s'effectue le codage des données? (ASCII, Word, documents structurés...)
- Modèles, nomenclatures, typologies :
    - Y a-t-il des modèles pour la structuration des documents (DTD, notices, formulaires, etc.) ?
    - Existe-t-il des modèles-« métier » (nomenclatures et référentiels) : lesquels ? Quel est leur degré de généricité ?
- Est-ce que l'utilisateur effectue des croisements entre bases et autres jointures « floues » ? Comment se font-ils ? Rencontre-t-il des problèmes? ?
- Y a-t-il de la collaboration entre les logiciels qu'il utilise : routines/ chaînes (cf. 1.a) ? Quels seraient ses souhaits ?
- Utilise-t-il les fonctionnalités suivantes : importation, exportation, réutilisation de données entre logiciels? Quelles sont les contraintes? les attentes ? Pratique-t-il des manipulations « maison »?

c. Traitement des résultats

- Qu'appelle-t-on « résultat » ? Est-ce le produit d'un traitement statistique ou un commentaire/ interprétation? Cela produit-il de nouvelles nomenclatures?
- Conserve-t-il les résultats intermédiaires ? lesquels ? et comment ?
- Quels sont les formats utilisés (cf. 1.b)? quel codage? quels modèles?
- Peut-on revenir aux corpus à partir des résultats ?
- Y a-t-il traçage des traitements? Si oui, existe-t-il de la documentation sur les procédures, les choix opérés ou le paramétrage du logiciel.
- Quels sont les critères de navigation dans les résultats? ?
- Y a-t-il exploitation « secondaire » et réutilisation de résultats ? de quel type ? avec quels outils ? Peut-on naviguer dans les bases ou les résultats ?
- Dialogue avec les autres services (ex. socio <--> marketing)
    - Quelles sont les contraintes pour le partage des résultats?

- Y a-t-il valorisation des résultats? Quels sont les types de résultats attendus ?

2. Remarques

- L'entretien sera semi-dirigé ; il n'est ni nécessaire, ni souhaitable que notre interlocuteur dispose du questionnaire à l'avance - nous gagnerons à recueillir d'abord son propre point de vue sur son travail, quitte à le compléter, en se rapportant à la grille.

- La présentation du projet sera adapté au métier des interlocuteurs (sociologie, datamining, marketing).

Annexe B. Guide d'installation

1. L'environnement Java

Afin de pouvoir utiliser le prototype, il est nécessaire d'installer soit un environnement de développement Java (SDK : Java2 Software Developpement Kit) soit un environnement d'exécution (JRE : Java2 Runtime Environment) qui permet à un utilisateur final l'exécution de programmes Java. Pour cela, il suffit de disposer d'une connexion internet et de se rendre sur le site de Sun Microsystems : http://java.sun.com/j2se/1.4/download.html

Une fois sur cette page, on choisit son système d'exploitation (Windows, Solaris ou Linux) et on télécharge le fichier d'installation.

Par exemple `j2re-1_4_2_04-windows-i586-p.exe` si on est sous Windows. Une fois le téléchargement terminé, il suffira de lancer cet "exécutable". Des menus vous guideront dans l'installation.

Une fois cette étape passée, on peut vérifier que l'environnement Java est bien installé en tapant dans un terminal la commande **java**. Si on obtient un message du type fichier non trouvé, il est nécessaire de mettre a jour votre PATH, cela s'effectue de la manière suivante :

- sous Unix/Linux

avec un terminal de type bash (ou un Émulateur Linux Cygwin)	Il faut éditer votre fichier .bashrc (situé à la racine de votre home directory) et rajouter a la fin : export PATH=$PATH:<répertoire d'installation du JRE>/bin export PATH=$PATH:.
avec un terminal de type tcsh	Il faut éditer votre fichier .tcshrc (situé à la racine de votre home directory) et rajouter a la fin : set PATH=$PATH:<répertoire d'installation du JRE>/bin set PATH=$PATH:.

- sous Windows

Windows 2000/NT ou XP	Il faut aller dans le menu :

	`Panneau de Configuration>Système>Avancés> Variables d'environnements` et rajouter dans ligne correspondant au PATH : répertoire d'installation du JRE>/bin;.
Windows 95/98	Il faut éditer votre fichier autoexec.bat (à la racine de votre unité C:\>) et rajouter : set PATH=%PATH%;<répertoire d'installation du JRE>/bin set PATH=%PATH%;.

Après ces modifications, il faut soit redémarrer la machine, soit mettre a jour l'instance du terminal.

- sous Unix/Linux

  ouvrir un nouveau terminal

  ou

  taper dans la ligne de commande du terminal courant :

  source ~/.bashrc ou **source ~/.tcshrc**

- sous Windows

  ouvrir un nouveau terminal

  ou

  relancer l'autoexec.bat dans le terminal courant :

  /autoexec

Après cela, la commande **java** devrait fonctionner.

2. Jena RDF Api

La Plateforme de Web Sémantique Jena est téléchargable sur http://jena.sourceforge.net/downloads.html

Il faut télécharger le fichier zip contenant l'intégralité de la Plateforme Jena. Une fois le téléchargement effectué, il faut décompacter ce zip dans un répertoire (dans la racine par exemple) et inclure dans le CLASSPATH les fichiers de la bibliothèque Jena :

```
export CLASSPATH=$CLASSPATH:~/Jena-2.1/lib/antlr.jar
export CLASSPATH=$CLASSPATH:~/Jena-2.1/lib/concurrent.jar
export CLASSPATH=$CLASSPATH:~/Jena-2.1/lib/jakarta-oro-2.0.5.jar
export CLASSPATH=$CLASSPATH:~/Jena-2.1/lib/junit.jar
export CLASSPATH=$CLASSPATH:~/Jena-2.1/lib/rdf-api-2001-01-19.jar
export CLASSPATH=$CLASSPATH:~/Jena-2.1/lib/xml-apis.jar
export CLASSPATH=$CLASSPATH:~/Jena-2.1/lib/commons-logging.jar
export CLASSPATH=$CLASSPATH:~/Jena-2.1/lib/icu4j.jar
export CLASSPATH=$CLASSPATH:~/Jena-2.1/lib/jena.jar
export CLASSPATH=$CLASSPATH:~/Jena-2.1/lib/log4j-1.2.7.jar
export CLASSPATH=$CLASSPATH:~/Jena-2.1/lib/xercesImpl.jar
```

3. VRP de ICS-Forth

Le validateur de ICS-Forth peut-etre trouvé à l'adresse suivante : http://139.91.183.30:9090/RDF/VRP

Après une inscription rapide (informations sur le futur utilisateur de VRP), on peut télécharger le fichier tar.gz, que l'on peut décompactera (avec Winzip sous Windows ou avec Ark sous Linux) dans un répertoire (par exemple la racine). Il faudrait comme précédemment inclure dans le CLASSPATH les fichiers suivants :

```
export CLASSPATH=$CLASSPATH:~/Rdf_vrp2_5/jars/JFlex.jar
export CLASSPATH=$CLASSPATH:~/Rdf_vrp2_5/jars/java_cup.jar
export CLASSPATH=$CLASSPATH:~/Rdf_vrp2_5/jars/relaxngDatatype.jar
export CLASSPATH=$CLASSPATH:~/Rdf_vrp2_5/jars/vrp2.5.jar
export CLASSPATH=$CLASSPATH:~/Rdf_vrp2_5/jars/xsdlib.jar
```

4. Le prototype

Le prototype est composé d'un seul fichier zip (`TEI.zip`). Il suffit de le décompresser son contenu dans un répertoire vide.

Il est nécessaire d'éditer le fichier `classpath` (`classpath.bat` sous Windows ou `classpath` sous Linux) car celui-ci fait référence aux librairies précédentes (Jena et VRP), il suffira juste de mettre à jour le nom des répertoires pour que Java sache où trouver les différentes librairies.

Le fichier **TEI** permet le lancement de l'application que l'on soit sous Windows ou sous Linux.

Bibliographie

Ouvrages

Shelley Powers. Pratical RDF - O'REILLY, Juillet 2003.

Madjid Ihadjadene, Helka Folch, Benoît Habert. Méthodes avancées pour les systèmes de recherche d'informations - Chap. 3 : Langages de métadonnées pour Web(s) sémantique(s) - HERMES, Mars 2004.

Elliotte Rusty Harold, W. Scott Means. XML in a Nutshell - 2nd edition - O'REILLY, Juin 2002.

Brett McLaughlin. Java et XML - O'REILLY, Mars 2002.

Norman Walsh, Leonard Muellner. Docbook, la référence - O'REILLY, mars 2001.

Sites Internet

Electricité de France division Recherche et Développement - http://www.edf.fr/index.php4?coe_i_id=20003

Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur (CNRS-Limsi) - http://www.limsi.fr

Extensible Markup Language (XML) - http://www.w3.org/XML

Resource Description Framework (RDF) - http://www.w3.org/RDF

Une introduction à RDF par l'Altruiste - www.laltruiste.com/coursxll/rdf.html

Le Web Sémantique - http://websemantique.org/PagePrincipale

Art Barstow - RDF Parsers - http://www.w3.org/People/Barstow/#online_parsers

The Java Tutorial - http://java.sun.com/docs/books/tutorial/index.html