République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université Ahmed Draia - Adrar

Faculté des Sciences et de la Technologie

Département des Mathématiques et Informatique

Mémoire de fin d'étude, en vue de l'obtention du diplôme de Master en informatique

Option : système informatique et site web dynamique

Thème

RETRO-INGÉNIERIE DES APPLICATIONS ORIENTÉES WEB À BASE D'ONTOLOGIES DE DOMAINE

Préparé par

Mr.Khaireddine Bousserhane

Encadré par

Mr. Mohamed Demri

Co-Encadreur

Mr. Djelloul Ben Atiallah

Année Universitaire**2015/2016**

Sommaire

Introduction générale

Avec l'arrivée de l'Internet et le Web, beaucoup d'applications ne sont plus développées en utilisant les technologies client/serveur traditionnelles. Au lieu de cela, de nouvelles applications sont développées, en l'occurrence, les applications Web. Une *application Web* est un système logiciel dont les fonctionnalités sont fournies à travers le Web. Les applications Web existantes peuvent avoir besoin de subir à un processus de rétro-ingénierie pour leur maintenance, évolution, migration ou compréhension. Pour satisfaire ces besoins, plusieurs approches ont vu le jour.

Pour couvrir la pauvreté sémantique des documents HTML sans réduire l'automatisation du processus de rétro-ingénierie, on propose dans ce mémoire une approche de rétro-ingénierie des applications Web à base d'ontologie de domaine pour générer un schéma conceptuel modélisant l'application Web.

L'objectif de notre approche est d'extraire *un sous schéma riche et réduit* décrivant l'application Web à base de l'ontologie de domaine. Le processus de rétro-ingénierie consiste en deux grandes phases : Tout d'abord, On doit *extraire les informations utiles* à partir des pages HTML (dans notre cas c'est des pages d'un site web touristique) pour les comparer par rapport aux informations présentées dans l'ontologie (ontologie touristique) ; Dans notre cas, les informations utiles sont celles présentées sous forme de tableaux et de listes, car c'est la forme la plus utilisée pour présenter des informations bien structurée dans une page Web. La deuxième étape est *l'identification* qui consiste à identifier ou à reconnaître les concepts de l'ontologie cachés dans les pages Web à l'aide des recherches sémantiques utilisées dans wordnet, puis inférer de nouveaux concepts et relations et enfin générer un schéma conceptuel UML (Ces deux dernières phases ne seront pas abordées dans notre mémoire).

Notre approche peut être utilisée pour la rétro-ingénierie des applications Web incluant des pages dynamiques avec un contenu qui se change continuellement. Elle est beaucoup plus orienté données, c.-à-d., qu'elle décrit l'aspect statique de l'application Web (Schéma de la base de données).

On propose aussi dans ce mémoire une série d'algorithmes pour la mise en œuvre des étapes du processus de rétro-ingénierie des applications Web à base d'ontologie.

Supportant l'approche proposée dans ce mémoire, on a développé un outil qu'on a nommé RetroWebOnto pour dire Rétro-ingénierie des applications Web à base d'ontologie.

- **Problématique**

Le World Wide Web (WWW), ou simplement le Web, est un framework architectural pour accéder à des documents liés et dispersés dans des milliers d'ordinateurs à travers Internet. Cependant, la façon aléatoire dont le Web s'est produit a eu une influence profonde sur son état actuel et en conséquence, une grande partie du Web existant est très difficile à maintenir et n'a pas été soumise à une maintenance systémique ou courante.

Les sites Web peuvent être vus comme des systèmes logiciels comportant des milliers de lignes de code HTML découpés en beaucoup de modules. Les pages statiques HTML sont une sorte de programmes avec des données encapsulées, qui manquent de performance par rapport aux sites dynamiques.

De nos jours, beaucoup de sites Web sont dynamiquement contrôlés. Les pages sont générées par des scripts (programmes) qui prennent les données à partir d'une base de données. La dissociation des données de la présentation peut surmonter ou simplifier les problèmes de maintenance de site Web comme les données désuètes, l'information inconsistante ou le style inconsistant.

Les applications Web sont souvent mal structurées et mal documentés, les principes d'ingénierie de logiciel sont rarement appliqués au développement des applications Web. Les développeurs originaux d'une application Web maintenue ne sont souvent plus une partie de l'organisation. La maintenance de tels systèmes est une problématique.

Les données sur le Web sont habituellement inclues dans les pages HTML, et elles ne correspondent pas à un schéma connu. Tandis qu'un utilisateur humain peut comprendre les données dans une page, il est impossible de faire ça par une machine.

Pour résoudre les problèmes décrits au-dessus, on a besoin du processus de la rétro-ingénierie des applications Web.

- **Structure du mémoire**

Pour la structuration de notre sujet, nous l'avons divisé en quatre chapitres :

Le 1^{ere} chapitre est consacré à décrire les ontologies et l'ingénierie des connaissances,

le 2eme chapitre présente un aperçu général sur les applications orientées web,

le 3eme chapitre montre qu'est le que la retro-ingénierie, les phases de cette approche ainsi que ses algorithmes et en fin

le 4^{éme} et dernier chapitre aborde l'implémentation et la réalisation de l'application.

I.1. Introduction.

Introduit en intelligence artificielle (IA) il y a 20 ans, le terme « ontologie » est cependant usité en philosophie depuis le XIXème siècle. Dans ce domaine, l'ontologie désigne l'étude de ce qui existe, c'est-à-dire l'ensemble des connaissances que l'on a sur le monde [WEL 01]. En IA, de façon moins ambitieuse, on ne considère que des ontologies, relatives aux différents domaines de connaissances. C'est à l'occasion de l'émergence de l'ingénierie des Connaissances que les ontologies sont apparues en IA, comme réponses aux problématiques de représentation et de manipulation des connaissances au sein des systèmes informatiques.

En informatique, une ontologie est un ensemble structuré de concepts. Les concepts sont organisés dans un graphe dont les relations peuvent être :

- Des relations sémantiques.
- Des relations de composition et d'héritage (au sens objet).

L'objectif premier d'une ontologie est de modéliser un ensemble de connaissances dans un domaine donné.

Les ontologies informatiques sont des outils qui permettent précisément de représentation un corpus de connaissances sous une forme utilisable par une machine.

I.2. Qu'est-ce qu'une ontologie ?[1]

L'ontologie est une notion difficile, complexe, à définir. Si l'on regarde dans le dictionnaire Le nouveau Petit Robert de la langue française 2007 [LE ROBERT 2006], on trouve la définition suivante : « PHILOS. Partie de la métaphysique qui s'applique à l'être en tant qu'être, indépendamment de ses déterminations particulières. » Il n'y a rien – sinon le nom – de prime abord, qui ne fasse référence à l'ontologie qui nous concerne et qui trouve sa place dans le domaine de l'ingénierie des connaissances. Comment cette notion a-t-elle glissé du domaine de la philosophie à celui de l'ingénierie des connaissances, pourquoi cette homonymie ? Nous tenterons dans un premier temps, de comprendre les racines de cette désignation avant de s'intéresser, dans un second temps, à ce qu'est l'ontologie en ingénierie des connaissances. Enfin, après ces propos, une question ne manquera pas de se poser : quelle est la place de l'ontologie au sein des langages documentaires ? Nous y apporterons donc, en dernier lieu, des éléments de réponses.

I.2.1. Un peu d'histoire.

Le mot ontologie est construit à partir des racines grecques : ontos qui veut dire « ce qui existe », « l'existant », et logos pour « le discours », « l'étude ». En d'autres termes, Ontologie signifie l'étude de ce qui existe, la science de l'être.

Pour tenter de définir cette notion d'ontologie, il n'est pas inutile de revenir brièvement sur les origines purement philosophiques de la notion et sur son histoire afin de comprendre pourquoi l'ingénierie des connaissances a choisi de réutiliser un mot préexistant pour désigner l'un de ses objets.

[1] http://memsic.ccsd.cnrs.fr/docs/00/35/59/20/PDF/Memoire_ontologie_Marcheix-vDefinitive.pdf

I.2.2. Du principe philosophique : la science de l'être en tant qu'être, à la création du terme d'Ontologie.

De manière à se constituer des fondements solides sur la notion d'ontologie, dans cette partie, nous nous intéresserons à son sens philosophique, le plus étroit et le plus théorique, où l'Ontologie est définie comme la théorie de l'être en tant qu'être.

Bien que le mot ontologie ne date que du XVIIe siècle, le principe peut en être trouvé chez Aristote dès le IIIe siècle avant-notre-ère. En effet, dans la philosophie classique, l'Ontologie correspond en partie à ce qu'Aristote appelait la Philosophie Première ou encore ce qui a été appelé la métaphysique, c'est-à-dire la science de l'être en tant qu'être, par opposition aux philosophies secondes – la physique notamment – qui, elles, s'intéressaient à l'étude des manifestations de l'être (les étant). La métaphysique peut se définir de la manière suivante : premièrement c'est une science qui étudie les premiers principes et les premières causes ; deuxièmement c'est une science qui étudie « l'être en tant qu'être » et non ce qui fait qu'un être est ceci ou cela mais ce qui fait qu'il est un être.

D'après d'Aristote, l'étant se dit de multiples façons. Cela signifie que l'être peut être divisé en catégories. Elles constituent les différentes descriptions associées aux manifestations de l'être dans le monde traduites par des propositions. Ainsi, on peut qualifier l'être selon dix catégories :

- Substance
- Quantité
- Qualité (quels attributs)
- Relation (plus …que, *etc*.)
- Lieu (où)
- Temps (quand)
- Posture (positionné comment)
- Possession (avec quoi)
- Action (en faisant quoi)
- Souffrance (subissant/étant affecté par)

Parmi ces dix catégories, la Substance a une importance prépondérante car elle constitue l'essence sans quoi une entité ne peut subsister, et qui par le fait même, individualise et différencie une entité par rapport à toutes les autres et elle assure une structure qui reste stable à travers les changements continuels du monde. Ainsi, il est possible de reconnaître un être (un certain individu par exemple), comme étant en essence le même, en dépit des changements qu'il subit de toutes les autres propriétés (les autres catégories dites accidentelles ou transitoires) au fil du temps, modifiant par exemple son apparence/qualité (nourrisson, enfant, adulte, vieillard), sa posture, ses actions, ses possessions, etc.

En outre, nous l'avons mentionné, le terme d'Ontologie ne date pas d'Aristote ou d'aucun autre de ses contemporains d'ailleurs, seul le principe est existant, et ce jusqu'au XVIIe siècle. En effet, le terme Ontologie naît en 1613 lorsqu'il apparaît, en grec, chez Goclenius (professeur de logique) et Lohrardus (recteur du Gymnasium de Saint-Gall) avant de se trouver sous sa forme latine dans les écrits de Clauberg (théologien et philosophe) en 1647 [CLAUBERG 1647].

Dans Lexicon philosophicum de Goclenius [GOCLENIUS 1613], le terme désigne la science de l'être en général. Il correspond par conséquent à la recherche en tant qu'être d'Aristote, parmi d'autres objets, propres à la philosophie première ou encore à la métaphysique.

Plus tard, Clauberg attribue la même signification au terme Ontologie dans Metaphysica et Ontosophie sive ontologia, où il l'emploie pour faire référence à une sorte de métaphysique générale qui aurait pour objet les caractéristiques essentielles communes à tous les êtres, à savoir : substance, existence, essence, etc.

Enfin, la diffusion du terme est due à l'Ontologia de Christian Wolff [WOLFF 1730], qui dans le concept scolaire de métaphysique, rangeait l'Ontologie en tant que métaphysique générale, puisqu'elle traitait de l'être en général, et la distinguait des trois sciences métaphysiques spéciales que sont la psychologie rationnelle (l'être de l'âme intellective), la cosmologie rationnelle (l'être du monde) et la théologie rationnelle (l'être de Dieu), chacune traitant d'une région déterminée de l'être.

I.2.2.1. Évolutions à travers le temps [PSYCHE et al. 2003].

À partir de cette première diffusion du terme, on peut dire que la notion d'Ontologie va faire l'objet de divergences, d'accords, d'enrichissements, en un mot évoluer. On a d'ailleurs pu noter qu'il y avait une différence entre la conception wolffienne de l'être et la conception classique. Cette différence dépendrait de ses prémisses leibniziennes qui veulent que le possible précède le réel, si bien que l'être est défini comme ce qui veut exister, soit qu'il existe effectivement, soit qu'il n'existe pas, l'existence apparaissant comme le complément de la possibilité.

Les principes suprêmes de l'Ontologie sont le principe de non-contradiction et le principe leibnizien de raison suffisante. Les déterminations internes de l'être sont ses attributs essentiels.

Pour le reste, l'Ontologie étudie une série de couples conceptuels, comme quantité et qualité, nécessité et contingence, simplicité et composition, finitude et infinitude, identité et diversité, cause et effet, etc.

Peu à peu, la notion d'Ontologie évolue. Kant conçoit son analytique transcendantale – première partie de la logique transcendantale, dans la Critique de la raison pure [KANT 2001] – de telle manière qu'elle peut prendre la place de la « vieille » Ontologie. Hegel agit de la même façon avec la logique qu'il identifie à la métaphysique. Il écrit dans l'un des textes introductifs à la Science de la logique : « la logique objective prend […] la place de la métaphysique d'autrefois » [HEGEL 1994].

Le terme d'Ontologie réinvestit le discours philosophique dans le cadre du développement de la phénoménologie. Il s'agit de la science des phénomènes, de la science de ce qui existe, de ce qui se manifeste à la conscience, soit directement, soit par l'intermédiaire des sens. Notamment, le projet de phénoménologie pure d'Husserl le conduit à parler d'ontologie régionale, ou sciences idéales de genres d'être qui empiriquement sont l'objet de plusieurs sciences – l'ontologie régionale de la nature physique par exemple. L'école existentialiste, avec Sartre, développe ensuite sa propre vision de l'Ontologie.

Dans la philosophie analytique, l'Ontologie a été étroitement liées à la logique et à la philosophie du langage. Selon Quine, les engagements ontologiques du discours ne sont pas tant déterminés par ses assertions[2] d'existence que par le type de variable sur lesquelles le langage admet la quantification : ainsi, une position nominaliste – pour qui il n'existe que des individus – admettra seulement la quantification sur des variables individuelles.

L'Ontologie est donc déterminée par la sémantique de son langage, et coïncide de fait avec les aspects généraux de cette sémantique. Un courant significatif de la philosophie analytique poursuit la construction d'une ontologie formelle, c'est-à-dire d'une théorie formelle des modes d'être. La construction d'une telle théorie coïncide avec la définition d'une sémantique pour un langage logique, dans laquelle peuvent trouver place les types d'entités que la théorie admet (par exemple, des individus, des classes ou bien des propriétés, etc.), et où sont définies les relations entre les différents types d'entités. Une telle ontologie formelle implique de soumettre à une reformulation dans le langage logique toutes les théories traditionnelles de l'être substantiel (idéalités mathématiques, réalités phénoménales des sciences naturelles, etc.).

À travers ce large panorama historique de l'Ontologie, nous avons pu remarquer le glissement d'une part de l'Ontologie de la théorie de l'être en tant qu'être à la méditation de l'être en général et d'autre part vers une formalisation dans un langage logique des théories traditionnelles de l'être substantiel. L'Ontologie s'interroge sur ce qu'est l'être, sur ce qui existe, sur ce qu'est la réalité, mais une question peut aller de pair avec tout cela : comment fait-on pour connaître, comment dire qu'une chose est ce qu'elle est ? On se demande ce que sont les choses, les étant, et comment faire pour les représenter ?

[2] Assertion : proposition, de forme affirmative ou négative, qui énonce un jugement et que l'on soutient comme vraie absolue. (TLFi – Trésors de la langue française informatisé)

I.3. L'ontologie en ingénierie des connaissances.

I.3.1. Qu'est-ce que l'ingénierie des connaissances ?[3]

L'ingénierie des connaissances (IC) ne porte pas directement sur les connaissances, car ces dernières ne sont pas des objets matériels sur lesquels effectuer des manipulations et transformations. L'IC porte sur l'inscription matérielle des connaissances, en se fondant sur le fait que :

Toute connaissance est l'interprétation d'une inscription qui en est l'expression, Toute inscription est matérielle et soumise à ce titre à une physique et peut faire l'objet d'une ingénierie.

L'IC ne porte pas sur toutes les inscriptions de connaissances, mais celles qui adoptent le support numérique comme substrat d'inscription. Le numérique confère une cohérence et une unité à l'ingénierie des connaissances :

Le support numérique est universel, et tout contenu peut s'inscrire sur un tel support, le support numérique est homogène au sens où les contenus inscrits peuvent être soumis à un même système technique.

L'ingénierie des connaissances sera donc l'ingénierie des inscriptions numériques de connaissances.

Elle élabore des outils, méthodes et dispositifs mobilisant les inscriptions numériques pour assister le travail de la pensée et l'exercice de l'esprit.

I.3.2. Définitions.

La réapparition et la diffusion du terme d'ontologie dans un domaine autre que la philosophie est un phénomène assez récent puisque qu'il date du début des années 1990.

En ingénierie des connaissances, les ontologies sont apparues dans le cadre des démarches d'acquisition des connaissances pour les systèmes à base de connaissances, les SBC. Les SBC ont succédé à ce qu'on appelle les systèmes experts, c'est-à-dire des outils capables de reproduire les mécanismes cognitifs d'un expert dans un domaine particulier. Plus précisément, un système expert est un logiciel capable de répondre à des questions en effectuant un raisonnement à partir de faits et de règles connus. Par rapport aux systèmes experts qui se composent d'une base de faits déclarative, d'une base de règles et d'un moteur d'inférence, les SBC proposaient de spécifier, d'un côté, des connaissances du domaine modélisé et, de l'autre, des connaissances de raisonnement décrivant les règles d'utilisation des connaissances du domaine. L'idée de cette séparation modulaire était de construire mieux et plus rapidement des SBC en réutilisant le plus possible des composants génériques, que ce soit au niveau du raisonnement ou des connaissances du domaine.

[3] http://web.iri.centrepompidou.fr/fonds/upload/seance/8/Museo-21_03_07-Ingenierie_des_connaissances.pdf

Ces dernières précisant tout ce qui a trait au domaine. Dans ce contexte, les chercheurs ont proposé de fonder ces connaissances sur la spécification d'une ontologie.

Par ailleurs, les ontologies s'inscrivent dans la continuité de nombreux travaux sur la représentation des connaissances, réseaux sémantiques, cartes et graphes conceptuels et on peut préciser que leur popularité dans le monde de la gestion de l'information a principalement bénéficié du développement du web sémantique [BERNERS-LEE 2001]. Le web sémantique ou web de données désigne un ensemble de technologies visant à rendre le contenu des ressources du Web accessible et utilisable par des programmes et agents logiciels grâce à un système de métadonnées formelles utilisant notamment les langages développés par le W3C. En d'autres termes, il s'agit d'une extension du Web actuel dans laquelle les informations, auxquelles on donne une signification bien définie, sont reliées entres elles de manière à ce qu'elles soient comprises par les ordinateurs dans le but de transformer la masse des pages Web en un index hiérarchisé et de permettre de trouver rapidement les informations recherchées.

Ainsi, dans un premier temps, on peut en déduire que le terme d'ontologie désigne des artefacts informatiques qui permettent la représentation d'un domaine de connaissance.

Pour aller plus loin, lorsque l'on tente de définir ce qu'est une ontologie, certains noms reviennent immanquablement comme notamment : Gruber, Sowa, Chandrasekaran, Uschold et Gruninger ou encore Guarino, parce qu'ils sont considérés comme des précurseurs et/ou théoriciens de l'ontologie. Même si actuellement de nouveaux noms font leur apparition dans le domaine, tels que Bruno Bachimont, Raphaël Troncy, Nathalie Aussenac-Gilles, pour entamer une définition des ontologies, on revient généralement toujours à ces débuts et aux premières personnes qui en ont parlé.

En effet, dans leurs écrits on peut constater que chacune d'elles apportent une pierre à la constitution d'une définition :

Pour Gruber, « une ontologie est une spécification partagée d'une conceptualisation ». [GRUBER 1993a]

Pour Sowa [SOWA 1999], une ontologie est un catalogue des types de choses supposées exister dans un domaine, du point de vue d'une personne utilisant un langage pour parler du domaine.

Pour Chandrasekaran [CHANDRASEKARAN et al.1999], une ontologie est une théorie du contenu sur les sortes d'objets, les propriétés de ces objets et leurs relations possibles dans un domaine spécifié de connaissances.

Pour Uschold et Gruninger [USCHOLD et GRUNINGER 1996], « il s'agit du terme utilisé se référant à la compréhension partagée d'un domaine d'intérêt qui peut être utilisé comme cadre unificateur pour résoudre les problèmes de communication entre les gens et d'interopérabilité entre les systèmes. »

Pour Guarino [GUARINO et POLI 1995] [GUARINO 1997a] [GUARINO 1997b], qui synthétise beaucoup d'éléments qui ont été dits sur les ontologies par d'autres personnes, il peut exister plusieurs ontologies concurrentes du même domaine, et la conceptualisation évoquée par Gruber, peut n'être que partielle. Dans une ontologie, il y a une distinction a priori entre les entités du monde, et entre les catégories utilisées pour modéliser le monde.

Ainsi, ces différents éléments, nous font comprendre ce que peut être une ontologie : une modélisation des connaissances ; un système, une théorie pour représenter un domaine de connaissances partagé par une communauté, etc. mais ne nous en donnent pas une définition complète et surtout ne laissent qu'entrevoir tout ce que la conception d'une ontologie implique, sans l'expliciter. Or, pour continuer ce mémoire et saisir son contenu, il faut se mettre d'accord sur une définition précise et rigoureuse. Pour ce faire, on utilisera les deux définitions complémentaires proposées par Jean Charlet, Bruno Bachimont et Raphaël Troncy dans un article datant de 2004 [CHARLET et al. 2004] :

Définition n°1 : Ensemble des objets reconnus comme existant dans le domaine. Construire une ontologie c'est aussi décider de la manière d'être et d'exister des objets.

Définition n°2 : Une ontologie implique ou comprend une certaine vue du monde par rapport à un domaine donné. Cette vue est souvent conçue comme un ensemble de concepts – e.g. entités, attributs, processus -, leurs définitions et leurs interrelations. On appelle cela une conceptualisation.

Une ontologie peut prendre différentes formes mais elle inclura nécessairement un vocabulaire de terme et une spécification de leur signification.

Une ontologie est une spécification rendant partiellement compte d'une conceptualisation.

La seconde définition propose un point de vue complémentaire. Il est cohérent avec la première mais plus précis, en termes de spécification et par rapport à une application informatique. Elle permet de préciser notamment les contraintes qui s'imposent successivement au concepteur d'une ontologie :

- une ontologie est une conceptualisation, c'est-à-dire que l'on y définit des concepts

- devant être utilisée par la suite dans un artefact informatique dont on veut spécifier le comportement, l'ontologie devra également être une théorie logique pour laquelle on précisera le vocabulaire manipulé

- la conceptualisation étant parfois spécifiée de manière très précise, une théorie logique ne peut pas toujours en rendre compte de façon exacte : elle ne peut assumer la richesse interprétative du domaine conceptualisé dans une ontologie et ne le fait donc que partiellement.

I.3.3. Les ontologies.

Ainsi, il est possible et même conseillé d'utiliser le pluriel pour parler de la notion d'ontologie afin de refléter les multiples facettes qu'elle recouvre.

Selon Van Heijst [VAN HEIJST et al.], on peut distinguer quatre types d'ontologies :

- Les ontologies du domaine : elles sont appelées de la sorte parce qu'elles expriment des conceptualisations spécifiques à un domaine. Elles rendent compte du vocabulaire d'un domaine spécifique au travers de concepts et de relations qui modélisent les principales activités, les théories et les principes de base du domaine en question. Elles sont réutilisables pour plusieurs applications concernant le domaine pour lequel elles ont été créées car elles ont été conçues de façon aussi indépendante que possible du type de manipulations qui vont être opérées sur ces connaissances.

- Les ontologies applicatives ou ontologies d'application : ce sont les ontologies les plus spécifiques, elles contiennent les connaissances requises pour une application particulière et ne sont pas réutilisables. Elles peuvent en outre inclure une ontologie de domaine.

- Les ontologies génériques ou ontologies de haut niveau : elles expriment des conceptualisations valables dans différents domaines de valeur relativement générale comme les notions d'objets, de propriété, de valeur, d'état, ou encore de temporalité. Théoriquement, ces ontologies doivent pouvoir être reliées au sommet des ontologies de domaines.

- Les ontologies de représentation : ce type d'ontologies regroupe les concepts utilisés pour formaliser les connaissances. Parmi les ontologies de représentation, on trouve des ontologies qui vont décrire les notions utilisées dans toutes les ontologies pour spécifier les connaissances, telles que les substances, les concepts, les relations etc. Par exemple, la « Frame-Ontology » est une ontologie de représentation. Elle définit de manière formelle les concepts utilisés principalement dans les langages à base de frames : classes, sous-classes, attributs, valeurs, relations et axiomes. Selon Guarino, les ontologies de représentation sont en fait indépendantes des différents domaines de connaissances, puisqu'elles décrivent des primitives cognitives communes aux différents domaines.

Par ailleurs, Uschold et Gruninger [USCHOLD et GRUNINGER 1996] précisent que les ontologies peuvent être de natures variables, suivant le type de langage utilisé et donc allant d'un degré de formalisation zéro à une formalisation totale. Quatre distinctions sont mises au jour :

- Les ontologies informelles : elles sont exprimées en langue naturelle. Ainsi, cela peut les rendre plus compréhensibles par l'utilisateur, mais cela rend plus difficile à vérifier l'absence de redondance ou de contradiction dans les ontologies. En d'autres termes, elles sont plus difficiles à valider.

- Les ontologies semi-informelles : elles sont exprimées dans une forme de langue naturelle structurée et limitée. Cela permet d'augmenter la clarté de l'ontologie tout en réduisant l'ambiguïté.

- Les ontologies semi-formelles : elles sont exprimées dans un langage artificiel et défini formellement.

- Les ontologies formelles : elles sont exprimées dans un langage artificiel disposant d'une sémantique formelle, permettant de prouver des propriétés de cette ontologie. L'intérêt de ces ontologies est la possibilité d'effectuer des vérifications sur l'ontologie : complétude, non-redondance, cohérence, etc.

Uschold et Gruninger expliquent également que du degré de formalisation de l'ontologie, dépend le degré d'automatisation dans les diverses tâches impliquant l'ontologie. « Si une ontologie est une aide à la communication entre personnes, alors la représentation de l'ontologie peut être informelle du moment qu'elle est précise et qu'elle capture les intuitions de chacun.

Cependant, si l'ontologie doit être employée par des outils logiciels ou des agents intelligents, alors la sémantique de l'ontologie doit être rendue beaucoup plus précise » [USCHOLD et GRUNINGER 1996].

Enfin, une dernière classification peut s'effectuer en fonction du niveau de granularité, c'est-à-dire du niveau de détail des objets de la conceptualisation. Ainsi, selon l'objectif opérationnel de l'ontologie, une connaissance plus ou moins fine du domaine est nécessaire et des propriétés considérées comme accessoires dans certains contextes peuvent se révéler indispensables pour d'autres applications. On peut relever alors deux types de granularités :

- Granularité fine : cela correspond à des ontologies très détaillées, possédant un vocabulaire riche capable d'assurer une description détaillée des concepts pertinents d'un domaine

- Granularité large : cela correspond à un vocabulaire moins détaillé. Les ontologies de haut niveau ont par exemple une granularité large, car les notions sur lesquelles elles portent peuvent être raffinées par des notions plus spécifiques.

À travers ces différentes typologies qui peuvent chacune selon leur critère de base (composition, nature et granularité) qualifier une ontologie nous voyons qu'il y a parfois contradiction avec la définition que nous avons choisie comme étant celle de référence pour ce mémoire. En effet, pour Jean Charlet, Bruno Bachimont et Raphaël Troncy l'ontologie doit pouvoir se prêter au raisonnement automatique. Il faut donc obligatoirement qu'elle soit conçue dans un langage formel. Or, nous venons de le voir, Uschold et Gruninger conçoivent quant à eux la création d'ontologies non formelles.

Cette divergence de position, parmi d'autres, par rapport à la conception d'une ontologie illustre le flou terminologique qui règne autour des ontologies. On note que la vision correspondant à la définition que nous avons choisie est plus répandue chez les « ingénieurs », que ce soit au sein de la communauté de l'intelligence artificielle ou de celle du web sémantique.

Néanmoins, les étapes de développement d'une ontologie permettent d'avancer une hypothèse pour articuler ces deux acceptations. Sans en dire trop sur le sujet car cela sera développé dans le Partie suivant, le processus d'élaboration d'une ontologie peut se schématiser selon Frédéric Fürst [FÜRST 2004] de la manière suivante :

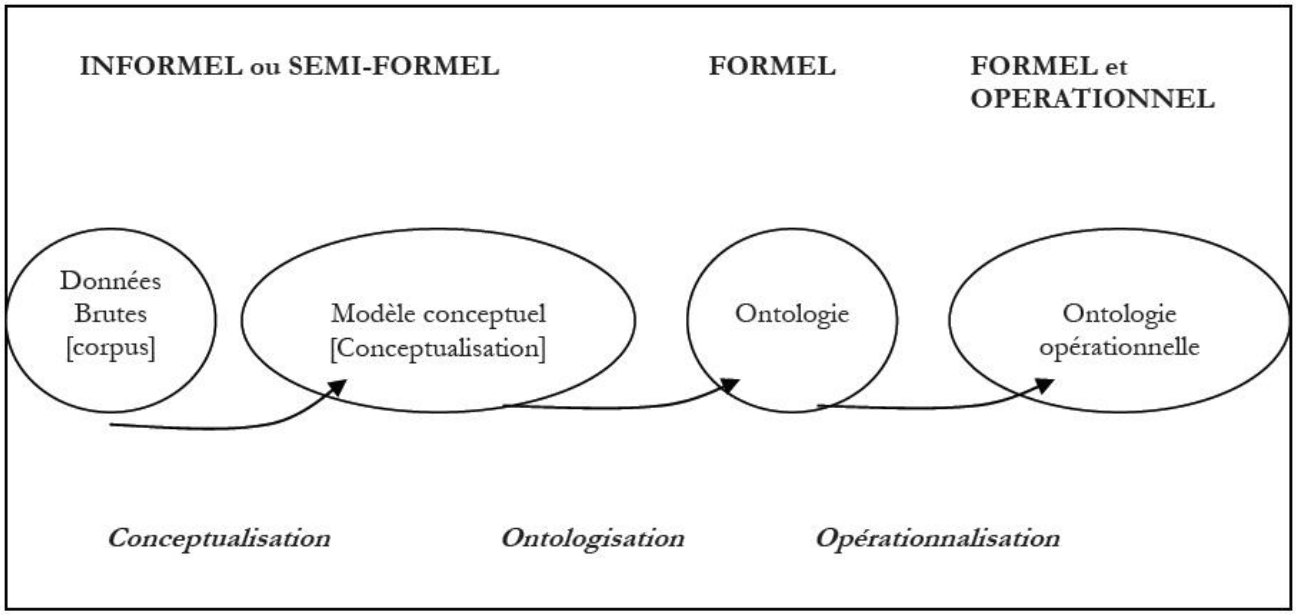


Figure I.3.3. : Construction d'une ontologie opérationnelle.

D'après cette figure, il est donc possible de concilier les deux visions. L'ontologie écrite en langage formelle est le produit fini mais il y a tout un chemin qui mène à cette formalisation.

I.3.3.1. Rôle des ontologies.[4]

La communauté d'intelligence artificielle utilise les ontologies pour deux raisons principales :

le partage et la réutilisation de connaissances, et l'amélioration de la communication.

Le partage est une des raisons les plus courantes qui conduit à développer des ontologies. Supposons, par exemple, qu'un certain nombre de sites contiennent de l'information sur un domaine. Si ces sites partagent et publient tous la même ontologie, qui est constituée des termes qu'ils utilisent, alors les agents informatiques peuvent extraire et agréger l'information de ces différents sites. La réutilisation des données sur un domaine précis est une des raisons majeures qui ont poussé la recherche sur les ontologies.

[4] http://www.lirmm.fr/~ducour/M2R/2006/Memoires/Rapport_M2R_07_Mbao.pdf

D'après [Ranwez, 2000] il existe trois types de communication dans un projet : communication homme-homme, homme-système ou entre les différents modules du système. Ces trois types de communication possèdent tous des caractéristiques particulières qui engendrent certains problèmes auxquels les ontologies peuvent apporter des solutions.

La communauté du Web sémantique utilise les ontologies pour diverses raisons. Elle l'utilise de façon simple pour améliorer la pertinence des recherches ; la requête exprimée peut ne faire référence qu'à un concept précis au lieu d'utiliser des mots-clés ambigus. Des applications plus avancées utilisent des ontologies pour associer l'information d'une page à des structures de connaissance et à des règles d'inférence.

Dans notre cas, nous allons visualiser les ontologies sur une carte conceptuelle. Cette carte doit servir de support à une recherche d'information dans de larges bases de connaissances. L'utilisateur doit cibler ses recherches sur les concepts proches de sa thématique (par exemple en biologie). Pour donner un sens à cette visualisation nous avons besoin d'une distance sémantique.

I.3.3.2. Quelle différence faites-vous entre une ontologie de l'ingénierie des connaissances et une ontologie philosophique ?[5]

On dit souvent que cela n'a pas de rapport. Il faut bien distinguer une ontologie avec un petit « o » et l'ontologie avec un grand « O », qui est celle étudiée par la philosophie. L'ontologie philosophique signifie la « Science de l'être en tant qu'être » tandis que les ontologies, avec un petit « o » et la marque du pluriel, sont les ontologies mobilisées dans les technologies de l'information et de la communication. Malgré tout on peut établir un lien dans la mesure où une ontologie, dans le contexte des ingénieries des connaissances, est l'énumération des concepts renvoyant à ce que l'on considère existé dans un domaine.

Vous avez à ce niveau deux manières de concevoir les ontologies avec « o » minuscule et pluriel : soit l'ontologie est un répertoire des objets du domaine on peut dire que c'est une vision réaliste, qui est souvent celle adoptée dans les travaux anglo-saxons sur les ontologies, soit, autre interprétation, l'ontologie ne désigne pas les objets du domaine mais renvoie à notre manière de penser le domaine. À ce niveau-là, il n'y a donc pas de prétention ontologique au sens philosophique du terme mais une prétention à fournir le terme nécessaire à la pensée dans le domaine.

[5] http://web.iri.centrepompidou.fr/fonds/upload/seance/8/Museo-21_03_07-Ingenierie_des_connaissances.pdf

La première approche est de déduire quels sont les objets qui existent dans ce domaine et donc à lister toutes les marques qui vous servent à désigner les objets d'un domaine. L'autre approche prend en compte la manière dont travaillent les spécialistes du domaine et fait le répertoire des notions dont ils se servent pour travailler dans ce domaine. Mais on ne s'occupe pas de savoir si ces notions renvoient à des objets ou pas et quel est le niveau de véracité de ces notions. C'est le travail des spécialistes de savoir si ce qu'ils disent est vrai ou pas. Dans l'ingénierie des connaissances, l'ontologie a simplement pour mission de fournir un répertoire ou un référentiel de concepts qui servira à soutenir et à sous-tendre les expressions de connaissance du domaine.

I.4. Comment construire une ontologie?[6]

Ce qui est paradoxal avec les ontologies, c'est que plus on essaye de savoir ce que c'est, plus on se perd. En effet, ce qui ressort incontestablement du premier chapitre c'est la polysémie du mot, et le flou terminologique qui en découle. Par souci de cohérence, nous avons opté pour une définition précise en deux parties proposée par Jean Charlet, Bruno Bachimont et Raphaël Troncy [CHARLET et al. 2004], qu'on rappelle ici :

Définition n°1 : Ensemble des objets reconnus comme existant dans le domaine. Construire une ontologie c'est aussi décider de la manière d'être et d'exister des objets.

Définition n°2 : Une ontologie implique ou comprend une certaine vue du monde par rapport à un domaine donné. Cette vue est souvent conçue comme un ensemble de concepts – e.g. entités, attributs, processus -, leurs définitions et leurs interrelations. On appelle cela une conceptualisation.

Une ontologie peut prendre différentes formes mais elle inclura nécessairement un vocabulaire de terme et une spécification de leur signification.

Une ontologie est une spécification rendant partiellement compte d'une conceptualisation.

En revanche, il n'existe pas de normes propres aux ontologies définissant tout le processus et les méthodologies de construction. Néanmoins, nous allons, dans le présent chapitre consacré à ce que l'on peut appeler l'ingénierie ontologique, tenter de le faire en donnant tout d'abord les grandes lignes directrices de la conception d'une ontologie, ce qu'elles doivent contenir.

[6] http://memsic.ccsd.cnrs.fr/docs/00/35/59/20/PDF/Memoire_ontologie_Marcheix-vDefinitive.pdf

I.4.1. Conception.

I.4.1.1. Constituants d'une ontologie.

À partir des définitions proposées jusque-là, et notamment à partir de celle de J. Charlet, B. Bachimont et R. Troncy, trois grands types de caractéristiques nous permettent de préciser ce qui peut être représenté dans une ontologie considérée en tant qu'objet informatique :

- Les concepts

- Les propriétés

- Les relations

L'hypothèse de départ, issue de la théorie des modèles[7] est qu'il existe des objets individuels qui peuvent être énumérés. Les concepts et les relations de l'ontologie sont organisés sous une forme hiérarchique qui admet une relation de subsomption[8].

I.4.1.1.1. Les concepts.

Selon François Rastier, chercheur au CNRS, « un concept est le signifié d'un mot dont on décide de négliger la dimension linguistique. Si bien qu'un concept n'est pas la source du terme, mais le produit de son instauration. C'est le travail terminologique qui transforme la notion en concept. Après avoir relié à une même notion plusieurs expressions jugées synonymes, on institue un concept en choisissant l'une d'entre elles comme terme, et en reléguant les autres au rang de pseudo-termes [RASTIER 1995]. ».

Les connaissances portent sur des objets auxquels on fait référence à travers des concepts

– dans certains travaux, on trouvera à la place de Concept, Classe. Un concept, selon Uschold et King [USCHOLD et KING 1995], peut représenter un objet matériel (par exemple une épée, un poignard, etc.), une notion (par exemple, la quantité) ou bien une idée.

[7] C'est une théorie de la vérité mathématique. Elle consiste essentiellement à dire qu'une théorie est mathématiquement valide si on peut définir un univers dans lequel elle est vraie.
[8] C'est une relation « est-un » ou encore « is-a »

Un concept peut se diviser en trois parties : un terme (qui se désigne sous le nom de Label), une notion et un ensemble d'objets. Le label d'un concept est l'expression linguistique utilisée couramment pour y faire référence. La notion désigne ce qui est appelé, au sens de la représentation des connaissances, l'intension d'un concept. Elle contient sa sémantique qui est définie à l'aide de propriétés, d'attributs, de règles et de contraintes. L'ensemble d'objets forme l'extension du concept. Il s'agit des objets auxquels le concept fait référence, autrement dit, ses instances[9]. Par exemple, le label du concept « pomme » considérée comme un fruit, renvoie aussi bien à la notion de pomme en tant qu'aliment que l'on mange, qu'à l'ensemble des objets de ce type : Golden, Cybel, Granny Smith, Elstar, etc.

I.4.1.1.2. Les propriétés.

Les propriétés – des attributs, dans le contexte des langages à objets – sont des caractéristiques liées aux concepts. Ainsi, une ontologie est non seulement le repérage et la classification des concepts mais c'est aussi des caractéristiques qui leur sont attachées. Ces caractéristiques peuvent être évaluées. Pour illustrer ce que sont les propriétés, J. Charlet, B. Bachimont et R. Troncy [CHARLET et al. 2004] prennent en exemple les sciences naturelles et leurs taxinomies[10] : les vertébrés ont un tégument (la peau) comportant des poils – pour les mammifères – ou des plumes – pour les oiseaux. Dans une ontologie sur le monde animal, on pourra avoir les concepts de « mammifère » ou d' « oiseau » pour lesquels est précisé le type de tégument, respectivement à poil et à plume. En pratique, un attribut « tégument » pourra être attaché aux concepts et sa valeur variera suivant le concept auquel on fait référence.

I.4.1.1.3. Les relations.

Les relations dans une ontologie représentent un type d'interaction entre les concepts, on peut en distinguer deux sortes :

- **La relation « est-un » :** cette relation de subsomption qui définit un lien de généralisation est utilisée pour structurer les ontologies. Elle permet fortement l'héritage de propriétés et c'est un choix qui s'impose depuis Aristote. Cette relation doit ensuite être complétée par d'autres relations pour exprimer la sémantique du domaine.

- Les autres relations : les relations unissent les concepts ensemble pour construire des représentations conceptuelles complexes. Si la connaissance ainsi construite correspond à un concept dans le monde modélisé, celui-ci est dit défini, à l'opposé des concepts insérés dans l'arborescence de l'ontologie qui sont dits primitifs. Par exemple, si l'on définit les lunettes comme étant des accessoires localisés sur le nez, c'est un concept défini. La relation « Localisés sur » est une relation binaire qui se définit par les concepts qu'elle relie et par le fait qu'elle est, comme les concepts, insérée dans une hiérarchie, une hiérarchie de relations cette fois-ci.

[9] Les instances d'un concept sont des éléments singuliers.

[10] Classifications dont la relation de base est une relation de subsomption.

La relation « est-un » est une relation de même type que les autres, mais elle a de spécifique que c'est elle qui a été choisie comme relation de structuration de l'arborescence ontologique. Elle est implicite dans une ontologie.

En ce qui concerne les autres relations et les concepts, J. Charlet, B. Bachimont et R. Troncy [CHARLET et al. 2004] insistent sur le choix du concepteur de l'ontologie. Ils tiennent à remarquer que les concepts et les relations de l'ontologie sont duals l'un par rapport à l'autre. Un concept primitif pourrait être un concept défini, une relation pourrait se retrouver implicitement définie au sein d'un concept primitif. Ce sont les choix assumés du concepteur de l'ontologie qui auront permis de décider de ce qui est essentiel – et donc primitif – ou non. Pour illustrer leurs propos, les trois auteurs donnent cet exemple : « on peut décider que le fait, pour un être humain, d'être un étudiant est temporaire donc non définitoire. On caractérise alors les êtres humains avec une relation de rôle social qui permettra de préciser une fonction d'étudiant ou de professeur. »

Enfin, un autre choix de conception doit être fait durant l'élaboration d'une ontologie : décider si une connaissance doit être modélisée dans une propriété ou à l'aide d'une relation pointant sur un autre concept. Une propriété peut être déclarée dès lors que les valeurs possibles sont d'un type dit primitif (entier, chaîne de caractères), et c'est une relation dès lors que les valeurs possibles sont d'un type dit complexe, c'est-à-dire un autre concept de l'ontologie. Néanmoins, cette frontière peut elle aussi être remise en question.

Ainsi, nous savons exactement de quoi doit se composer une ontologie : de concepts, de propriétés, et de relations. Mais d'autres questions se posent alors : comment créer l'ontologie, comment passer de la théorie à la pratique, y a-t-il des techniques particulières pour collecter les concepts si nous ne les avons pas, comment les classer, comment éviter au maximum les erreurs de modélisation, de structuration, etc. ? Autrement dit, existe-il une ou des méthodes pour créer une ontologie ? La réponse à cette question est Oui. Il existe en effet quelques méthodes ou éléments de méthode dans le domaine. Cependant, (par manque de temps certes), mais aussi parce que le but de l'exercice n'est pas de faire un relevé de tout ce qui existe dans le domaine de l'ingénierie ontologique, mais de trouver le juste milieu entre théorie et pratique, j'ai fait le choix de ne présenter qu'une méthodologie de construction d'ontologies, celle que j'ai utilisé pour réaliser ma mission.

I.4.1.2. Engagement sémantique et engagement ontologique pour la conception et réalisation d'ontologies [BACHIMONT 2000].

L'ingénierie des connaissances participe à rendre les ontologies interprétables par une machine. Elle est chargée de la « modélisation ontologique » – le fait de définir les primitives de représentation et leur signification qui seront utilisées pour la modélisation formelle des connaissances –, étape préalable à la modélisation formelle, c'est-à-dire, à la représentation dans un langage formel des connaissances du domaine.

Elle comprend plusieurs étapes permettant de passer de l'expression linguistique des connaissances telles que nous autres, humains, pouvons la considérer à une représentation formelle et calculable des connaissances propres à une exploitation informatique. Cette transformation se fait par l'intermédiaire de deux engagements, l'un sémantique, et l'autre ontologique. En outre, elle permet notamment de résoudre, ou du moins propose des pistes pour résoudre, les problèmes soulevés précédemment au sujet du rôle du concepteur de l'ontologie par rapport au choix des primitives, ce qui relève de l'essentiel ou non.

I.4.1.2.1. 1ère étape : extraction de termes et analyse.

L'un des premiers objectifs à atteindre lorsque l'on construit une ontologie est de définir les primitives du domaine, tout en sachant qu'il n'existe pas de primitives dans un domaine. Les primitives, comme les termes de tête d'un thésaurus, sont arbitraires. Il faut par conséquent, modéliser les primitives nécessaires à la formalisation et à la représentation du problème à résoudre et des connaissances s'y rapportant. La construction des primitives relève du choix du concepteur, mais comment procéder ? Pour y parvenir, Bruno Bachimont propose de repartir de l'expression linguistique des connaissances du domaine en utilisant l'extraction de termes à partir d'un corpus spécialisé. En effet, un corpus (constitué de documents propres au domaine concerné) comporte l'expression des notions qu'il faut modéliser. On peut ainsi construire un corpus textuel qui sera la source privilégiée permettant de caractériser les notions utiles à la modélisation d'une ontologie et le contenu sémantique qui lui correspond. Pour ce faire, B. Bachimont utilise ce qu'il appelle « une démarche corpus » et des outils terminologiques pour commencer la modélisation du domaine. Ces outils, pour la plupart, reposent sur la recherche de formes syntaxiques particulières manifestant les notions recherchées comme des syntagmes nominaux pour des candidats termes, des relations syntaxiques marqueurs de relations sémantiques, ou des proximités d'usage – comme les contextes partagés – pour des regroupements de notions.

En ce qui nous concerne, cette première étape n'est pas obligatoire puisque nous disposons déjà, à travers le thésaurus PACTOLS des candidats termes[11]. En revanche, cette étape peut être un avantage pour un pré-classement des termes et relations.

[11] Le travail de regroupement de termes et élaboration des concepts avait déjà été réalisé en 1987 pour la création des PACTOLS. 46 000 mots-clés ont été organisés, regroupés et hiérarchisés pour aboutir à 2500 descripteurs thématiques.

I.4.1.2.2. 2ème étape : la normalisation sémantique.

À la fin de la première étape, nous avons donc une liste de candidats termes dont les libellés ont un sens pour le spécialiste du domaine. Mais rien n'assure que ce sens soit unique : au contraire argumentent Jean Charlet, B. Bachimont, et Raphaël Troncy [CHARLET et al. 2004] car nous sommes dans un fonctionnement linguistique où les significations sont ambigües, les définitions circulaires dépendent en particulier du contexte interprétatif des locuteurs. Or, dans la modélisation ontologique, on cherche à construire des primitives dont le sens ne dépend pas des autres primitives et est surtout non contextuel. Il faut dès maintenant prendre le chemin du formel en normalisant les significations des termes pour ne retenir, pour chacun d'eux, qu'une seule signification, qu'une seule interprétation par un être humain. C'est ce que permet l'utilisation de la sémantique différentielle, proposée par Bruno Bachimont. Cette sémantique, issue notamment des travaux de François Rastier [RASTIER 1987] [RASTIER et al. 1994], permet de décrire les unités entre elles par les identités qui les unissent et les différences qui les distinguent.

Cela donne lieu à une définition de l'unité selon quatre principes différentiels :

- Le principe de communauté avec le père : toute unité se détermine par l'identité qu'elle possède avec l'unité parente. Il faut expliciter en quoi l'unité fille est identique à l'unité parente. Il s'agit du principe aristotélicien de définition par le genre proche.

- Le principe de différence avec le père : toute unité se distingue de l'unité parente, sinon il n'y aurait pas lieu de la définir. Il faut donc expliciter la différence qui la distingue de l'unité parente. Il s'agit du principe aristotélicien de définition par la différence spécifique.

- Le principe de différence avec les frères : toute unité se distingue de ses frères, sinon il n'y aurait pas lieu de la définir. Il faut donc expliciter la différence de l'unité avec chacune des unités sœurs.

- Le principe de communauté avec les frères : toutes les unités filles d'une unité parente possèdent, par définition, un même trait générique, celle qu'elle partage avec l'unité parente. Mais il faut établir une autre communauté entre unités filles : celle qui permet de définir des différences mutuellement exclusives entre les unités filles. Bruno Bachimont donne cet exemple : l'unité parente est être humaine, et les unités filles sont homme et femme.

Ces unités partagent le fait d'être des humains. Mais cette propriété ne permet pas de définir en quoi les hommes et les femmes sont différentes. On choisit alors comme principe de communauté la sexualité, on peut attribuer à homme le trait masculin, et à femme le trait féminin. Ces deux traits sont mutuellement exclusifs, car ce sont deux valeurs possibles d'une même propriété.

À la fin de cette étape, nous avons un arbre de primitives conceptuelles valable dans la seule région du monde modélisée où les concepts retenus correspondent bien à ceux de l'ontologie, par définition décontextualisée. Nous avons ce que B. Bachimont appelle une ontologie régionale.

I.4.1.2.3. 3ème étape : l'engagement ontologique.

L'engagement ontologique correspond à l'évolution de l'ontologie régionale vers une ontologie formelle. La sémantique formelle ne considère plus des notions sémantiques mais des extensions, c'est-à-dire l'ensemble des objets qui vérifient les propriétés définies en intension dans l'étape précédente, propriétés ayant une définition formelle à ce niveau. La structure est alors un treillis.

Ce treillis de concepts doit être vu comme la possibilité de créer des concepts dits défini en combinant les concepts primitifs.

I.4.1.2.4. 4ème étape : l'opérationnalisation.

Il s'agit de la dernière étape de la création d'une ontologie, elle est généralement le fruit du travail d'un logiciel. Elle débouche sur la manipulation de l'ontologie par une machine. Elle devient un objet informatique. Enfin, cette étape consiste en la représentation de l'ontologie dans un langage de représentation de connaissances, ce qui nous amène directement à nous intéresser aux principaux langages de connaissances utilisés pour la création des ontologies formelles.

I.4.2. Langages de représentation des connaissances.

Audrey Baneyx, dans sa thèse Construire une ontologie de la pneumologie : aspects théoriques, modèles et expérimentation [BANEYX 2007], donne trois exigences auxquelles le langage de représentation des ontologies doit se soumettre :

- La lisibilité : le langage doit être compréhensible pour un utilisateur humain et doit donc avoir une certaine continuité avec le langage naturel

- La portabilité : le langage choisi doit être le plus standard possible afin de pouvoir être réutilisé dans d'autres systèmes

- La possibilité de pouvoir faire des inférences : le langage doit permettre le traitement informatique des données en vue de calculer les déductions logiques possibles

Par ailleurs, un langage d'ontologie doit permettre de signifier l'appartenance d'un objet à une catégorie, de déclarer la relation de généralisation entre catégories et de typer les objets que lie une relation.

Autrement dit, un langage de représentation pour les ontologies doit être capable de représenter toutes les subtilités de l'ontologie (concepts, propriétés et relations) et à l'heure d'Internet où les échanges sont démultipliés, mais aussi simplement par soucis d'interopérabilité et de normalisation, il se doit d'être le plus standard possible, et aussi le plus utilisé. En ce qui concerne la première exigence – la lisibilité – tout est relatif : si l'on n'est pas coutumier des langages de représentation en général, un langage de représentation pour les ontologies ne sera pas plus lisible que les autres.

Un langage répond actuellement à ces critères, il s'agit d'OWL (Ontology Web Language) issu des travaux sur le web sémantique dirigés par le W3C. Ce langage occupe une place prépondérante dans le paysage des ontologies et est le standard le plus utilisé, c'est pourquoi nous en ferons la description dans un premier temps, avant de nous intéresser à un autre type de langage de représentation des connaissances : SKOS (Simple Knowledge Organisation System), qui, comme son nom l'indique est utilisé pour représenter de manière moins lourde les schémas de concepts.

I.4.2.1. OWL.

OWL[12], est issu d'un groupe de travail dédié au développement de langages standards pour modéliser des ontologies utilisables et interchangeables. Il a acquis le statut de recommandation du W3C le 10 février 2004. C'est un dialecte XML, fondé sur le standard RDF comme on peut le voir sur la figure suivante. RDF et OWL, standards du web sémantique, fournissent un cadre de travail pour la gestion des ressources, l'intégration, le partage et la réutilisation des données sur le Web.

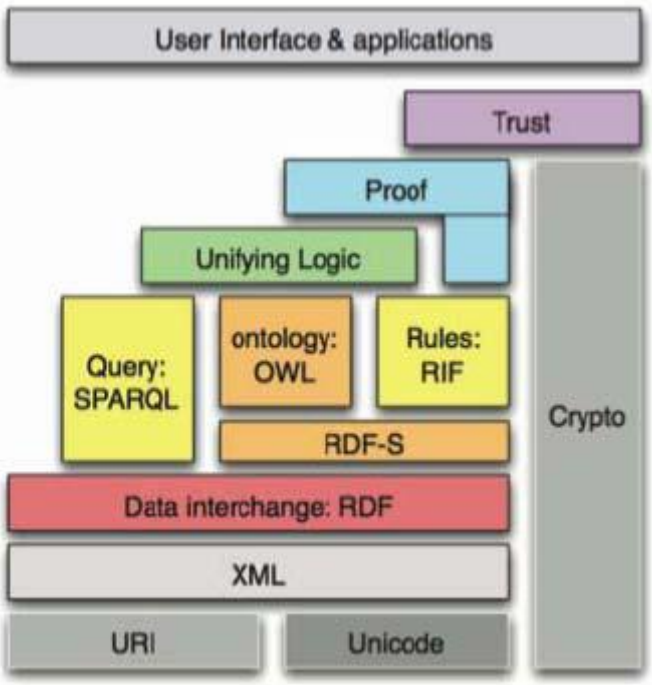


Figure I.4.2.1.a : Le « gâteau » de Tim Berners-Lee extrait du support de l'exposé de Raphaël Troncy donné à l'occasion d'une journée d'étude sur le Web Sémantique en juillet 2008.

La vision du Web sémantique dans lequel l'information serait accessible et manipulable automatiquement par la machine, s'appuie sur une pile de langages jouant chacun un rôle particulier :

- XML fournit une manière de représenter des documents structurés, mais il n'impose aucune contrainte sémantique sur les documents produits ;

[12] Owl signifie chouette, symbole de Minerve, déesse de la sagesse et de la guerre.

- RDF est un modèle de données simple, fondé sur des ressources et des relations entre ces ressources, équipé d'une sémantique et qui peut se représenter en XML ;

- RDF Schéma permet de définir le vocabulaire pour décrire des classes et des propriétés hiérarchisées en taxinomies ;

- OWL fournit davantage de primitives de modélisation pour décrire des ontologies plus riches sur le web.

Voici une présentation rapide de la structure, intéressons nous maintenant plus en détail à ces différents langages, en partant de XML jusqu'au principal concerné : OWL

XML[13]

XML (eXentesible Markup Language) est un langage informatique de balisage générique. Il est recommandé par le W3C pour exprimer des langages de balisage spécifique, tels que RDF ou OWL. Par rapport au langage HTML (HyperText Markup Language), XML laisse la possibilité à son utilisateur de distinguer les données selon leur sens et leur contenu. Un document XML se présente sous la forme de données « *taggées* » par un ensemble de balises, chacune pouvant comporter des attributs et des valeurs. Il n'y a pas de définition figée des balises. Son objectif initial est de faciliter l'échange automatisé de contenus entre systèmes d'information hétérogènes – on parle alors d'interopérabilité.

Il s'agit de ce qu'on peut appeler la « première couche » d'OWL.

RDF[14]

RDF (Resource Description Framework) est un modèle de représentation sémantique des informations du Web qui utilise la syntaxe XML. Il permet la mise en place de descriptions simples sur les ressources du Web comme les auteurs de pages Web, leur date de création, etc. Les ressources du Web sont l'élément de base de RDF. Chaque ressource est pourvue d'un identifiant uniforme de ressource (URI, Uniform Resource Identifier). Initialement recommandé par le W3C dans le but de standardiser les définitions et les usages des métadonnées, RDF est également utile à la représentation de données elles-mêmes.

La structure fondamentale de toute expression en RDF est une collection de triplets, chacun composé d'un sujet, d'un prédicat et d'un objet. Un ensemble de tels triplets est appelé un graphe RDF.

[13] http://www.w3.org/XML/

[14] http://www.w3.org/RDF/

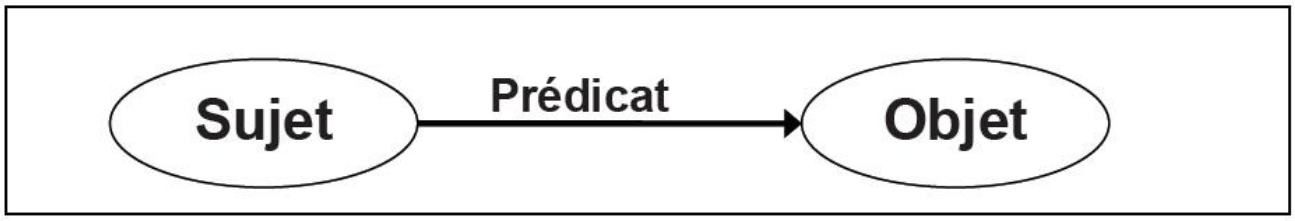


Figure I.4.2.1.b : Représentation d'un triplet RDF.

Dans un graphe, chaque triplet représente l'existence d'une relation entre les choses symbolisées par les nœuds qui sont joints.

Les schémas RDF (RDFS[15]) permettent de définir le vocabulaire utilisé dans les descriptions RDF. Il confère un formalisme de représentation riche incluant des classes, sous-classes, propriétés, sous-propriétés, des règles d'héritages de propriétés, etc., mais ne normalise pas les inférences que l'on pourrait faire avec. La structure objet-classe des RDFS permet de représenter un modèle du domaine en définissant des objets du domaine et leurs relations pour rendre compte d'une ontologie.

OWL

OWL[16] (Web Ontology Language) doit permettre de représenter des ontologies, en particulier sur le Web. Il est fondé sur la syntaxe RDF/XML et est dédié totalement à la représentation des ontologies.

OWL est destiné à être utilisé lorsque les informations contenues dans les documents doivent être traitées par des applications logicielles, c'est-à-dire lorsqu'elles ne sont pas simplement montrées à l'utilisateur. Une ontologie OWL est composée d'un en-tête (métadonnées), d'axiomes et de faits. Les axiomes concernent la définition complète ou partielle de concepts et de relations, la spécification de propriétés sur les relations (propriétés algébriques) et la définition d'axiomes sur les classes et les relations (équivalences, expression booléenne). Parmi les relations, on distingue celles dont le domaine de valeur sera de type primitif (attribut) de celles dont le domaine de valeur sera un autre concept (relation). Les faits concernent des individus pour lesquels on donne des valeurs aux propriétés des classes dont ils sont les instances.

Il existe trois sous-langages d'OWL offrant des capacités d'expression croissantes : OWL Lite, OWL DL et OWL Full.

- OWL Lite est le sous langage d'OWL le plus simple. Son principal intérêt est de permettre la modélisation d'ontologies simples, d'une complexité formelle peu élevée, de sorte qu'il soit facile d'implémenter des raisonneurs corrects et complets.

[15] http://www.w3.rog/TR:rdf-schema/

[16] http://www.w3.org/2004/OWL/

- OWL DL est plus complexe qu'OWL Lite. Il permet une expressivité plus importante. Il est fondé sur la logique descriptive (d'où son nom, OWL Description Logics), un domaine de recherche étudiant la logique, lui conférant son adaptation au raisonnement automatisé. Il garantit la complétude des raisonnements (toutes les inférences sont calculables) et leur décidabilité (leur calcul se fait en une durée finie).

- OWL Full est la version la plus complexe d'OWL, mais également celle qui permet le plus haut niveau d'expressivité. Il est destiné aux situations où il est plus important d'avoir un haut niveau de capacité de description, quitte à ne pas pouvoir garantir la complétude et la décidabilité des calculs liés à l'ontologie. OWL Full propose néanmoins la possibilité d'étendre le vocabulaire par défaut d'OWL.

Enfin, il existe entre ces trois langages une dépendance de nature hiérarchique : toute ontologie OWL Lite valide est également une ontologie OWL DL valide, et toute ontologie OWL DL valide est également une ontologie OWL Full valide.

Il existe certes d'autres langages de représentations d'ontologies, mais OWL est celui qui tend à s'imposer aujourd'hui. En outre, c'est un standard du W3C. C'est-à-dire qu'à défaut d'être une norme ISO, il en est l'équivalent dans le monde des industriels.

I.4.2.1.1. Structure d'une ontologie OWL. [17]

- Espaces de nommage : L'espace de noms, parfois appelé espace de nommage permet d'indiquer avec précision de quel vocabulaire les termes d'une ontologie proviennent. C'est la raison pour laquelle, comme tout autre document XML, une ontologie commence par une déclaration d'espace de noms contenue dans une balise rdf:RDF.

Exemple : le laboratoire LGI2P est partenaire du projet TOXNUC-E qui regroupe plus de 200 chercheurs autour de la toxicologie nucléaire (projet MNRT, CEA, INSERM, CNRS, INRA). Les exemples (ToxNucFev07.owl) choisis sont tirés de l'application de nos travaux à ce projet.

```
<rdf:RDF
  xmlns     = "http: /www.owl-ontologies.com/ToxNucFev07#"
  xml:base  = "http: /www.owl-ontologies.com/ToxNucFev07">
  xmlns:owl = "http: /www.w3.org/2002/07/owl#"
  xmlns:rdf = "http: /www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs= "http: /www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd = "http: /www.w3.org/2001/XMLSchema#">
</ rdf:RDF>
```

La première déclaration identifie l'espace de nommage propre à l'ontologie que nous sommes en train d'écrire (ToxNucFev07). La deuxième déclaration identifie l'URI de base de l'ontologie. Les quatre dernières déclarations introduisent le vocabulaire d'OWL et les objets définis dans l'espace de nommage de RDF, du schéma RDF et des types de données du Schéma XML.

[17] http://www.lirmm.fr/~ducour/M2R/2006/Memoires/Rapport_M2R_07_Mbao.pdf

- En-tête d'une ontologie : L'entête décrit le contenu de l'ontologie courante. La balise owl:Ontology permet d'indiquer les informations contenues dans l'ontologie.

Eléments de base

Il existe divers éléments de base pour le langage OWL, mais nous allons décrire ceux qui nous semblent les plus importants pour mettre en Œuvre notre distance. Pour des explications exhaustives du rôle de chacun des éléments composant le vocabulaire d'OWL, il est recommandé de se reporter à la Recommandation du W3C (OWL Web Ontology Language Reference) [W3C 2004].

- Classes : Une classe ou concept est définie comme un groupe d'individus qui ont des caractéristiques similaires dans un domaine. Le langage OWL dispose d'une superclasse nommée owl:Thing. Chaque classe définie par l'utilisateur est donc implicitement une sous- classe de owl:Thing. Il existe également une classe nommée noThing, qui est sous-classe de toutes les classes OWL. Ces deux concepts permettent de construire le treillis des concepts.

Voici un exemple de déclaration de classe : <owl:Class rdf:ID="Biologie">

L'appel de la classe Biologie au sein du document se fait par #Biologie, par exemple rdf:resource='#Biologie'. La propriété subClasseOf permet de relier une classe spécifique (BiologieStructurale) à une classe plus générale (Biologie). Exemple:

```
<owl:Class rdf : ID ="BiologieStructurale">
   <rdfs:subClassOf rdf:resource="# Biologie" />
</owl>
```

- Propriétés : Dans le langage OWL, une propriété permet de définir des faits ou des relations entre des classes. Il existe en OWL deux types de propriétés : propriétés d'objet (owl : ObjectProperty) qui définissent une propriété entre deux individus d'une classe ou de plusieurs classes et les propriétés de type de données (DataTypeProperty) qui relient des instances à des valeurs de données).

- Instances de classe : L'ensemble des individus d'une classe est désigné par le terme extension de classe, chacun de ces individus étant alors une instance de la classe. Les instances sont utilisées pour représenter les éléments spécifiques.

I.4.2.2. SKOS.

Le projet de SKOS[18] (Simple Knowledge Organisation System) a été initié par l'Union européenne dans le cadre du projet SWAD-Europe[19]. Ces travaux ont abouti en 2003 aux premières publications de SKOS Core Guide et SKOS Mapping Guide, ainsi qu'à une mise en application destinée à valider les technologies retenues en situation réelle. La réflexion a ensuite été reprise par le W3C dans le cadre du groupe de travail sur les bonnes pratiques et le déploiement des standards RDF.

Il s'agit d'un langage de représentation de schémas de concepts. Comme son nom l'indique, il est destiné à proposer un système permettant d'exprimer et de gérer des modèles interprétables par des machines dans la perspective du web sémantique. On le dit « simple » par opposition notamment au langage OWL, qui comme nous l'avons vu est plus à même de représenter des structures plus riches : les ontologies. En effet, SKOS est avant tout destiné à la représentation de thésaurus, classifications, listes de vedettes matières, taxinomies ou autres folksonomies.

Son formalisme de représentation repose sur les graphes RDF. Le concept constitue le centre du graphe auquel peuvent notamment être attachés en tant que propriétés RDF :

- Les indications portant sur le concept lui-même : des termes préférentiels ou alternatifs, les équivalents dans d'autres langues ; des termes cachés ; et la représentation par une image

- Les différents types de notes : notes de définition et d'application, exemples, notes historiques, etc.

- Les relations sémantiques : hiérarchie et association

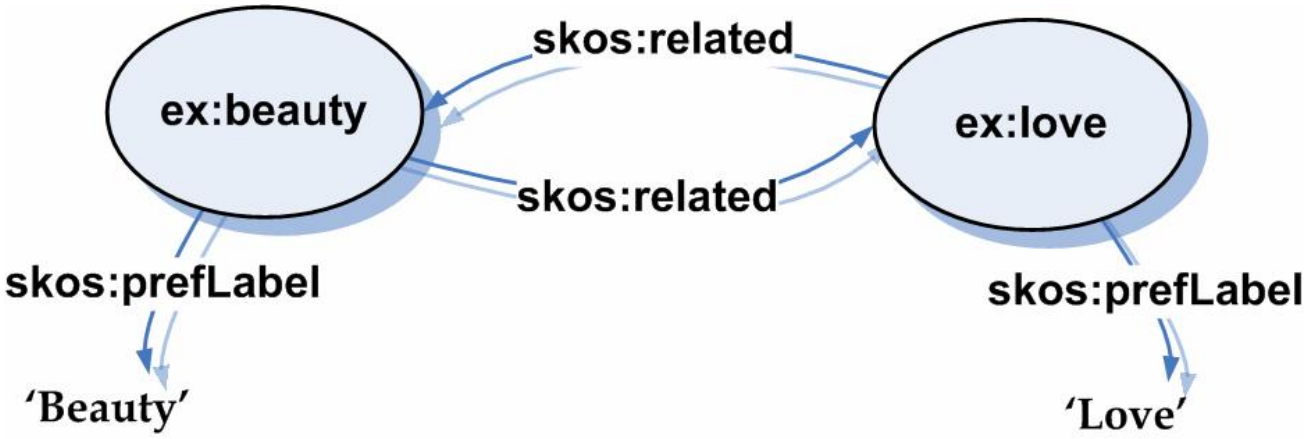


Figure I.4.2.2 : Représentation de la relation associative dans SKOS.

[18] http://www.w3.org/2004/02/skos/

[19] Semantic Web Advance Development for Europe

I.4.3. Éditeurs d'ontologies.

```
<owl:Class>
 <owl:oneOf rdf:parseType="Collection">
  <owl:Thing rdf:about="#Eurasia"/>
  <owl:Thing rdf:about="#Africa"/>
  <owl:Thing rdf:about="#NorthAmerica"/>
  <owl:Thing rdf:about="#SouthAmerica"/>
  <owl:Thing rdf:about="#Australia"/>
  <owl:Thing rdf:about="#Antarctica"/>
 </owl:oneOf>
</owl:Class>
```

```
<owl:Class>
 <owl:intersectionOf rdf:parseType="Collection">
  <owl:Class>
   <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Tosca" />
    <owl:Thing rdf:about="#Salome" />
   </owl:oneOf>
  </owl:Class>
  <owl:Class>
   <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Turandot" />
    <owl:Thing rdf:about="#Tosca" />
   </owl:oneOf>
  </owl:Class>
 </owl:intersectionOf>
</owl:Class>
```

Figure I.4.3 : Syntaxe OWL.

Voici deux exemples de syntaxe en OWL, qui comme nous l'avons vu dans la partie précédente, est le langage standard pour représenter une ontologie. C'est donc sous cette forme que doit être transformée l'ontologie pour devenir interprétable par une machine, ultime étape. Mais pour ceux qui n'ont pas le temps d'écrire toutes ces lignes … ou pour ceux, et j'en fais partie, qui n'éprouvent aucun besoin particulier de s'essayer à l'apprentissage de l'écriture de tels langages, il existe fort heureusement des logiciels qui prennent en charge l'écriture des ontologies dans le langage formel de notre choix, ce sont des éditeurs d'ontologies.

Les éditeurs d'ontologies jouent donc un rôle d'intermédiaire entre le concepteur de l'ontologie et le langage informatique.

I.4.3.1. Protégé.[20]

Protégé a été développé par le Stanford Medical Informatics de l'université de médecine de Stanford depuis 1995. Il est construit autour d'un modèle de connaissances inspiré par le paradigme des frames : classes, slots (attributs) et facets (contraintes sur les attributs) qui sont les primitives de modélisation proposées. Ce modèle autorise une liberté de conception importante, puisque le contenu des formulaires de spécification des classes peut être modifié suivant les besoins, via un système de méta-classes, qui constituent des sortes de « patrons » pour les classes du modèle du domaine. Il est adapté à la construction d'ontologies depuis la version Protégé 2000. L'interface très complète ainsi que l'architecture logicielle bien pensée permettant l'insertion de plugins, notamment des plugins pour gérer les représentations sous forme graphique, par exemple OWLViz, ont grandement contribué au succès de Protégé. En quelques années, cet éditeur s'est imposé comme la référence, avec une communauté d'utilisateurs extrêmement importante et active. Ses nombreuses extensions lui permettent en particulier de gérer des langages standards comme RDF et surtout OWL, de créer des axiomes formels de manière intuitive, d'accéder aux ontologies par des interfaces graphiques évoluées, de comparer et fusionner des ontologies avec la suite PROMPT[21]. Il est également possible de faire fonctionner des raisonneurs, comme RACER[22] (Renamed ABox and Concept Expression Reasonner) pour le langage OWL par exemple, pour vérifier la cohérence et la consistance de la structure ontologique.

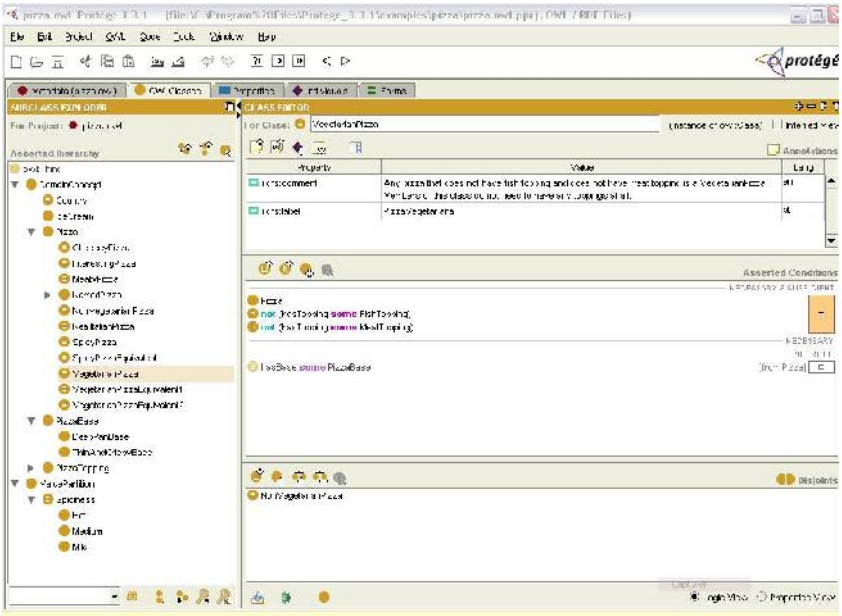


Figure I.4.3.1 : Copie d'écran de l'interface principale de l'éditeur d'ontologies Protégé.

[20] Appelé auparavant Protégé 2000, cet éditeur a repris le nom de l'outil d'acquisition des connaissances qui l'a précédé. http://protege.stanford.edu./

[21] http://protege.cim3.net/cgi-bin/wiki.pl?Prompt

[22] http://www.racer-systems.com/products/racerpro/index.phtml

I.4.3.2. OilEd.

L'éditeur OILED[23] a été développé par l'université de Manchester pour éditer des ontologies dans les langages de représentations OIL, puis DAML+OIL, les précurseurs d'OWL. Il est donc explicitement orienté vers la représentation en logique de description expressive et, à ce titre, fournit tous les éléments d'interface permettant de spécifier des hiérarchies de concepts et de rôles, ainsi que la construction des expressions complexes définissant ces entités. À l'origine il n'a pas d'autre ambition que d'illustrer les vertus du langage pour lequel il a été créé. Les versions disponibles d'OiLEd ne constituent pas un environnement complet pour le développement d'ontologies d'envergure. En effet, cet outil n'implémente pas la migration et l'intégration d'ontologies, ne gère pas les différentes versions et autres activités impliquées dans la construction d'ontologies. Néanmoins, la simplicité, la robustesse de cet outil et la présence d'un raisonneur logique de description FaCT[24], capable de tester la faisabilité des ontologies construites ou d'expliciter de nouvelles relations de subsomption entre concepts complexes, en font un outil de référence relativement populaire avec plus de 2000 téléchargements. Comme le soulignent les concepteurs, il s'agit plutôt d'un « bloc-notes » offrant assez de fonctionnalités pour permettre à des utilisateurs de construire les ontologies et en assurer l'uniformité.

OILED permet d'exporter des ontologies construites dans des langages standards tels que DAML+OIL, RDFS ou OWL.

I.4.3.3. WebODE.

WebODE[25] est une plateforme en ligne développée par le groupe Ontological Engineering du département d'Intelligence artificielle de la faculté d'Informatique de l'université polytechnique de Madrid. Elle se place au niveau méthodologique dans la lignée d'ODE, un éditeur qui assurait le support de Methontology, la méthodologie proposée par ce laboratoire. L'ambition nouvelle de WebODE par rapport à ODE est de considérer que les ontologies doivent être construites et mises à disposition via le web pour faciliter le développement d'application du web sémantique.

WebODE est composé de plusieurs modules : un éditeur d'ontologies qui intègre la plupart des services nécessaires à la construction d'ontologies (édition, navigation, comparaison, fusion, raisonnement…), un système de gestion des connaissances à base ontologique, un générateur automatique de portail du web sémantique, un outil pour annoter les ressources du web et un éditeur de services pour le web sémantique. La plateforme WebODE met l'accent sur la possibilité d'un travail collaboratif et sur la possibilité, comme dans Protégé, d'étendre la plateforme à l'aide de modules complémentaires, comme un moteur d'inférences ou bien l'outil ODEClean, […], accepte l'export et l'import d'ontologies en RDFS, DAML+OIL et OWL.

[23] http://oiled.man.ac.uk/

[24] FaCt Classification of Terminologies, http://www.cs.man.ac.uk/horrocks

[25] http://webode.dia.fi.ump.es/WebODEWeb/index.html

I.5. Conclusion.

Nous nous sommes intéressés à ce qu'était une ontologie. Pour cela nous sommes partis des origines philosophiques du terme pour ensuite définir son sens en ingénierie des connaissances.

Ensuite, nous avons étudié la manière de concevoir et de réaliser une ontologie en ingénierie des connaissances en en énumérant ses composants et en proposant une méthode de construction.

Le langage de représentation le plus répandu, standard du W3C, pour représenter des ontologies, OWL, et d'autre part, en exposant les différents logiciels permettant d'éditer des ontologies.

II.1. Introduction. [1]

En informatique, une application Web (aussi appelée site Web dynamique ou WebApp) est un logiciel applicatif manipulable grâce à un navigateur Web. De la même manière que les sites Web, une application Web est généralement placée sur un serveur et se manipule en actionnant des widgets à l'aide d'un navigateur Web, via un réseau informatique (Internet, intranet, réseau local, etc.).

Les Webmails, les systèmes de gestion de contenu, les wikis, les blogs sont des applications Web.

Les moteurs de recherches, les logiciels de commerce électronique, les jeux en ligne, les logiciels de forum peuvent être sous forme d'application Web.

[1] http://fr.wikipedia.org/wiki/Application_Web

Des appareils réseau tels par exemple les routeurs sont parfois équipés d'une application Web dans leur micro logiciel.

Les applications Web font partie de l'évolution des usages et de la technologie du Web appelée abusivement Web 2.0.

II.2. Histoire.[2]

Le World Wide Web est un système de documentation hypertexte créé en 1993 pour les besoins du Centre européen pour la recherche nucléaire (CERN). Le premier navigateur Web (NCSA Mosaic) a été créé la même année par le National Center for Supercomputing Applications (NCSA).

Le World Wide Web a permis aux utilisateurs de se partager des documents et des images plus rapidement que via le courrier électronique et plus facilement que via le partage de fichiers.

Le nombre grandissant de documents publiés a rendu rapidement les moteurs de recherche nécessaires pour les retrouver : l'utilisateur entre un mot clé, le serveur Web effectue la recherche, puis envoie le résultat sous forme d'un document.

Les moteurs de recherche ont été mis en œuvre par extension du serveur Web.

En 1995, le NCSA publie la norme industrielle CGI, qui spécifie quelles sont les modalités d'extension d'un serveur Web, dans le but de le brancher avec un logiciel applicatif - par exemple un moteur de recherche.

La technologie des applications Web a évolué très rapidement entre 1994 et 2000, où plusieurs logiciels de serveurs Web réservés à cet usage sont sortis sur le marché un après l'autre, ainsi que des améliorations des navigateurs Web.

Les premières applications Web souffraient d'une maniabilité et d'une ergonomie inférieure aux applications client. Est appelée Rich Internet Application une application Web qui offre une maniabilité et une ergonomie équivalente à une application client. Le terme Rich Internet Application est apparu la première fois dans une publication de Macromedia en 2002.

II.3. Serveur informatique.[3]

Un serveur informatique, ou serveur lorsque le contexte s'y prête, est l'un des éléments participant au mode de communication client-serveur entre des logiciels: un logiciel dit « client » envoie une requête à un logiciel « serveur » qui lui répond, le tout suivant un protocole de communication.

[2] http://fr.wikipedia.org/wiki/Application_Web

[3] http://www.techno-science.net/?onglet=glossaire&definition=3811

Par extension, on désigne par serveur informatique l'ordinateur hébergeant de tels logiciels serveurs. Les logiciels clients s'y connectent à travers un réseau informatique. Les serveurs offrent des services qui permettent, par exemple, de stocker des fichiers, transférer le courrier électronique, héberger un site Web, etc. Il est possible pour un ordinateur ou un logiciel d'être client et serveur en même temps.

La connexion client-serveur utilise des protocoles de communication, comme par exemple TCP/IP, qui est le protocole le plus utilisé sur l'Internet.

Différents types de serveurs :

– Serveur d'applications.

– Serveur web.

– Serveur de base de données.

– Serveur de fichiers.

– Serveur d'impressions.

II.3.1. Serveur d'applications. [4]

Les serveurs d'application sont des logiciels qui aident des entreprises à développer, déployer et contrôler un grand nombre d'applications qui sont la plupart du temps distribuées. Du point de vue développement, la différence principale apportée par un serveur d'application est la séparation du logique métier de la logique présentation et de la logique base de données. Essentiellement, les serveurs d'application nous aident à démontrer des applications 3-tiers où la base de données est logiquement séparée (parfois physiquement séparé aussi) du logique métier.

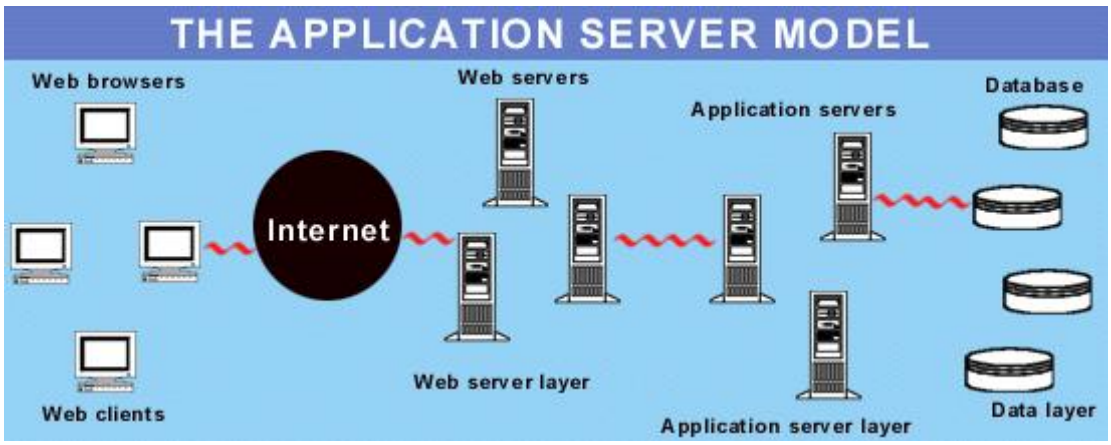


Figure II.3.1.a. : Modèle de serveur d'application.

[4] www710.univ-lyon1.fr/~exco/.../serveur_application.pdf

Développer un serveur implique de traiter beaucoup de questions compliquées :

- Accès concurrents (simultanéité)
- Permission d'accéder à toutes les bases de données possibles de production
- Gestion de connexion réseau
- connexion à base de données
- Interface de gestion
- Équilibrage de charge
- Tolérance aux pannes
- Extensibilité de votre travail et performance

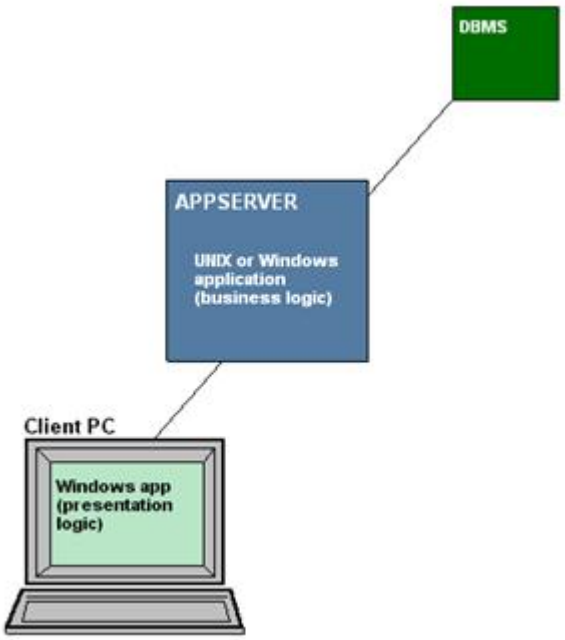


Figure II.3.1.b. : Un serveur d'application dans un environnement de client/serveur 3-tiers fournissant un processus de traitement entre la machine de l'utilisateur et le système de gestion de base de données (SGBD)

Un serveur d'application peut simplifier votre procédé de développement. Les serveurs d'application prennent habituellement soin de la plupart, sinon de toutes les questions techniques impliquées, et permettent aux développeurs de se concentrer sur la raison première du projet. Ceci permet de budgéter pour des systèmes beaucoup plus grands et beaucoup plus utiles.

II.3.2. Serveur Web.[5]

De simple relais de pages HTML statiques, les serveurs web ont évolué pour remplir des tâches complexes de messagerie, de commerce électronique, ou de partage d´information. L´offre hétérogène des produits impose d´examiner les critères qui garantiront la cohérence du site, et ses éventuelles évolutions, avec l´environnement système.

Les serveurs web ont pour fonction de relayer l´information vers les clients (navigateurs Internet Explorer ou Netscape) internes (Intranet) ou externes (Internet, Extranet). Ces logiciels participent de l´architecture client/serveur, fondée sur le protocole de communication HTTP et le langage de publication HTML. Le terme de serveur web, parfois utilisé de façon étendue pour désigner les machines sur lesquelles fonctionnent ces programmes, ne concerne que la partie software.

II.3.2.1. Quelle configuration ?

La machine sur laquelle tourne le serveur web est choisie pour son importante quantité de mémoire vive, celui-ci devant répondre à un maximum de requêtes clients en un minimum de temps. Le système d´exploitation dépend du type de site, statique ou dynamique. Un site statique est compatible avec tout système d´exploitation, le serveur web ayant pour unique fonction la publication des pages HTML.

Un site dynamique impose l´existence d´un serveur web avec des modules d´interprétation de scripts et de bases de données. Lorsque les bases de données représentent plusieurs centaines de Mo, un serveur spécifique de bases de données devient indispensable. Malgré la possibilité de passerelles multi plateformes, le système d´exploitation impose pour partie le choix du serveur web. Certains systèmes d´exploitation fonctionnent étroitement avec un langage de scripts : le PHP avec le système UNIX, l´ASP avec le système Windows. De même, pour les serveurs de base de données : Unix avec les bases de données MySQL, et Windows avec Microsoft Access ou SQL Server.

II.4. Application Web.[6]

Une application web est une extension dynamique d'un serveur web. Une application web est formée d'un ensemble de composants web, de ressources statiques (images, sons, …), de bibliothèques et de classes utilitaires.

Les composants web fournissent cette capacité d'extension. Les servlets et pages jsp sont des exemples de composants web.

Les composants web sont supportés par un container web (tomcat) qui leur fournissent un ensemble de services tels que les services de sécurité, de concurrence, de cycle de vie, ….

[5] http://www.indexel.net/article/serveurs-web-les-criteres-de-choix.html

[6] http://deptinfo.cnam.fr/new/spip.php?pdoc3392

La configuration d'une application web en vue de son déploiement est maintenue dans un fichier XML, appelé descripteur de déploiement.

II.4.1. Les applications Web statique.[7]

Au début du Web, les pages HTML se limitaient à l'affichage de simples textes et à quelques illustrations (dont l'affichage était d'ailleurs souvent bloqué pour améliorer la fluidité de la navigation sur les réseaux à faible débit de l'époque). Au fil des années, avec l'avènement d'Internet tel qu'on le connaît, les exigences des utilisateurs ont évolué. En effet, la seule interactivité possible sur les pages HTML était l'affichage une nouvelle page lors d'un clic sur un lien hypertexte. Il était donc impossible d'obtenir le moindre effet visuel (comme un simple roll-over sur une image par exemple) sans avoir recours à une technologie complémentaire. De plus, l'envoi d'une requête au serveur Web, suite à un clic sur un lien hypertexte par exemple, engendrait un cycle de traitement long et fastidieux qui freinait considérablement la réactivité des applications sur des réseaux et des serveurs souvent sous-dimensionnés pour le trafic sans cesse croissant de l'époque.

II.4.1.1. Les sites statiques.

Les sites statiques sont constitués d'un ensemble de pages HTML reliées entre elles par des liens hypertextes qui permettent de naviguer de l'une à l'autre. Le protocole utilisé pour transférer des informations Web sur Internet s'appelle HTTP. Une requête HTTP (http://www.eyrolles.com/page.htm par exemple) est envoyée vers le serveur afin d'accéder à la page désirée et de la visualiser dans le navigateur du poste client (voir étape 1 de la figure II.4.1.1*)*.

Lorsque le serveur Web reçoit cette requête, il recherche la page demandée parmi toutes les pages HTML présentes sur le site concerné et la renvoie ensuite au client (voir étape 2 de la figure II.4.1.1). Le code HTML reçu par le poste client est alors interprété et affiché par le navigateur (voir étape 3 de la figure II.4.1.1). C'est ce qu'on appelle l'architecture client-serveur (je demande, on me sert) : le client est le navigateur Internet (Internet Explorer, Firefox...) et le serveur est le serveur Web sur lequel sont stockées les informations du site Internet. Ce type de site est très simple à réaliser et la plupart des premiers sites ont été conçus sur ce modèle. Cependant, ce concept est limité car il manque cruellement d'interactivité.

[7] www.numilog.net/package/extraits_pdf/e258970.pdf

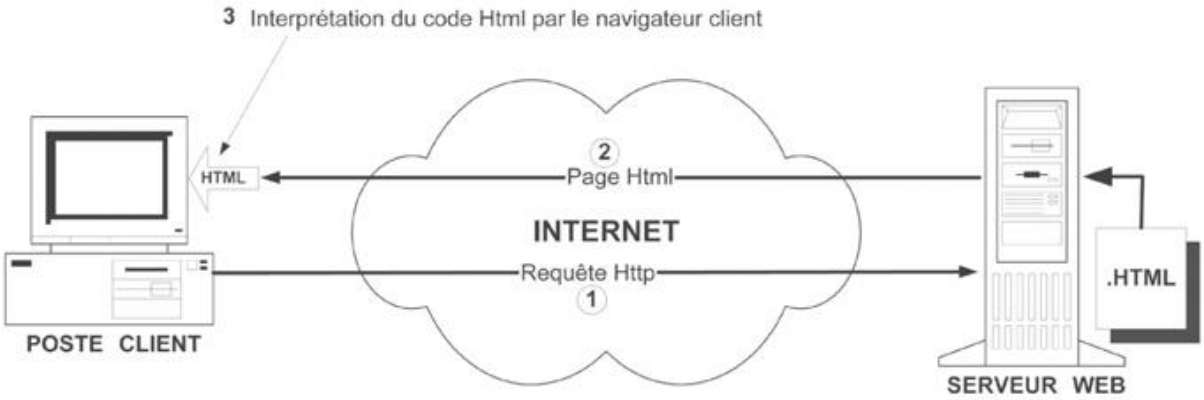


Figure II.4.1.1 : Les sites statiques.

L'architecture client-serveur d'un site statique : le poste client envoie au serveur une requête HTTP ; le serveur Web recherche puis fournit au poste client la page demandée, qui est ensuite interprétée par le navigateur.

II.4.1.2. À la découverte du protocole http.[8]

HTTP (HyperText Transfer Protocol) est un des principaux protocoles du web, destiné à l'échange de documents reliés entre eux par des liens hypertextes. Il s'agit d'un protocole de la couche application du modèle OSI, et basé sur les protocoles plus bas niveau TCP et IP. Le client se connecte au serveur, lui envoie une requête (en général il demande le contenu d'un fichier html), éventuellement accompagnée de données brutes (le contenu d'un formulaire, par exemple) ; le serveur lui répond en lui renvoyant d'abord des en-têtes de réponse avant de lui envoyer le contenu du fichier. Il s'agit d'un protocole sans état : si le client effectue plusieurs requêtes successives auprès d'un même serveur, le serveur n'a pas de moyen de s'en apercevoir. Ainsi, lors de l'accès à une page protégée, les identifiants sont transmis à chaque requête, même si l'utilisateur final ne les entre qu'une fois.

II.4.1.2.1. La requête.

Une requête HTTP est constituée de 4 sections :

– Une ligne de requête, précisant le service désiré, une ressource (un fichier) ainsi que la version HTTP utilisée, généralement HTTP/1.1.

– Des headers, un par ligne, qui servent à apporter des informations facultatives au serveur : les langues acceptées par le client, le type du navigateur, des cookies, les types de compression acceptées, etc.

[8] graal.ens-lyon.fr/~mgallet/files/asr2/dm2.pdf

– Une ligne vide, pour signaler la fin des headers.

– Un corps de requête optionnel, contenant par exemple le formulaire. HTTP/1.1 définit 8 types de requêtes différents :

– GET : demande au serveur de lui envoyer la ressource précisée.

– HEAD : identique à GET, sauf que seuls les en-têtes seront transmis.

– POST : identique à GET, sauf que des informations supplémentaires seront transmises dans le corps de la requête.

– PUT : envoie une ressource au serveur.

– DELETE : efface une ressource sur un serveur.

– TRACE : le serveur renvoie au client sa propre requête.

– OPTIONS : permet au client de connaître les méthodes supportées par le serveur.

– CONNECT : transforme la requête en tunnel TCP/IP (pour permettre les communications HTTPS).

Voici un exemple de requête :

```
GET /index.php HTTP/1.1
Host: www.example.com
```

II.4.1.2.2. La réponse.

La réponse d'un serveur est également constituée de 4 sections :

– La première précise à nouveau le protocole utilisé, ainsi qu'un code de retour (200 = OK, 404 = NOT FOUND, etc.).

– Des headers, avec le même rôle que pour la requête.

– Une ligne vide, pour signaler la fin des headers.

– Le corps de la réponse.

Voici un exemple de réponse :

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
```

Etag: "3f80f-1b6-3e1cb03b"

Accept-Ranges: bytes

Content-Length: 438

Connection: close

Content-Type: text/html; charset=UTF-8

II.4.1.3. Le XHTML.[9]

L'eXtensible HyperText Markup Language est un des langages qui servent à rédiger des pages Internet. Comme son nom l'indique, il s'agit d'un langage à balises. Les balises sont de la forme <balise>. Une balise peut avoir des attributs. Dans ce cas, on écrira :

<balise attribut1="valeur1" attribut2="valeur2"... >

Une page XHTML est un fichier texte, avec .html comme extension. Une page XHTML commence par <html> (on ouvre la balise) et se termine par </html> (on ferme la balise). En XHTML, toutes les balises ouvertes doivent être refermées. Suivant le principe des poupées russes, la dernière balise ouverte sera la première à être fermée. Il est conseillé d'écrire la balise fermante en même temps que la balise ouvrante. Une page XHTML comporte deux parties : les entêtes et le corps. La première commence par <head> et se termine par </head>. La deuxième est, elle, balisée avec <body> et </body>.

Pour faire du XHTML 1.0 Strict, on mettra au tout début de la page :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
```

[9] http://www.eleves.ens.fr/home/levieil/html/html.pdf

II.4.1.3.1. La balise standard.

Voilà un petit tableau des principales balises :

Balise	Fonction	Remarques
<title>	Titre de la page	Dans les en-têtes
<h1>	Titre interne	Il y a aussi <h2>, <h3>... pour les titres(headings) moins importants
<p>	Paragraphe	
<a href="...">	Lien	Anchor
	Liste	Unordered List
	Element d'une liste	List Item
<table>	Tableau	
<tr>	Ligne de tableau	Table Row
<td>	Element d'un tableau	Table Data
<img src="...">	Image	Ne pas oublier l'attribut alt

Figure II.4.1.3.1. : La balise standard.

II.4.1.3.2. Quelques attributs.

Liens L'attribut href permet de désigner la cible du lien. Il est possible d'écrire href="mailto:nom@domaine.fr" pour envoyer des mails, mais attention au spam. L'attribut name permet de donner un nom à l'ancre, ce qui permet de faire un lien vers cette ancre précisément, en utilisant

href="http://nom_de_la_page#nom_de_l'ancre".

Images L'attribut alt permet de décrire le contenu de l'image, il est obligatoire, et utile pour l'accessibilité de vos pages. L'attribut src permet d'indiquer l'emplacement de l'image. Deux autres attributs très utiles sont height et width, qui permettent d'indiquer respectivement une hauteur et une largeur pour l'image.

Tableaux Par défaut, les tableaux ont une bordure d'épaisseur nulle. En écrivant border="1", on peut donner à la bordure une épaisseur d'un pixel.

Meta On peut mettre une ou plusieurs balises <meta> dans les en-têtes.

name="author" content="Moi"	Nom de l'auteur
http-equiv="Content-Language" content="fr"	Langue de la page
name="keywords" lang="fr" content="mot_clé1,mot_clé2"	Mots-clés

II.4.1.3.3. Valider ses pages Web.

Respecter les normes est le meilleur moyen de s'assurer que votre page Web s'affichera correctement sur tous les navigateurs. Il existe plusieurs validateurs, par exemple celui du W3C : http://validator.w3.org/. Pour montrer que vous avez pris soin de respecter les normes, vous pouvez ajouter le code suivant sur vos pages :

```
<p>
  <a href="http://validator.w3.org/check?uri=referer"><img
    src="http://www.w3.org/Icons/valid-xhtml10"
    alt="Valid XHTML 1.0 Strict" height="31" width="88" /></a>
</p>
```

En outre, en cliquant dessus, vous pourrez vérifier la conformité de votre page. Il existe aussi un validateur pour le CSS(cf. section 2) : http://jigsaw. w3.org/css-validator/

II.4.2. Les applications Web dynamique.[10]

Un site Web dynamique est un site Web dont les pages sont générées dynamiquement à la demande. Le contenu est obtenu (par exemple) en combinant l'utilisation d'un langage de scripts ou de programmation et une base de données. Il s'agit souvent de PHP pour le langage et MySQL pour la base de données. Dans les sites dynamiques, le contenu (articles) est séparé de l'habillage (modèles ou squelette). Cette séparation contenu/présentation/logique est le credo des développements actuels. Les avantages sont donc loin d'être négligeables, et les possibilités de dynamisation évoluent de jour en jour. Les rédacteurs du contenu ne sont pas forcément habilités à publier leurs articles. L'administrateur quant à lui peut valider ou non les articles et changer l'habillage.

II.5. Les composants d'une application Web interactive.[11]

Les composants classiques d'une application Web interactive sont: l'architecture, le réseau d'entreprise, la communication via Internet, la sécurité et la mise en œuvre.

II.5.1. L'architecture.

La définition de l'architecture, en fonction des besoins et objectifs d'une entreprise, est le premier pas dans l'élaboration d'une solution. Dans le monde des technologies de l'informatique et des télécommunications, le terme architecture définit à la fois les directions à prendre pour la réalisation de projets à court, moyen ou long termes, la structure générale des solutions à mettre en œuvre ainsi que le cadre des travaux à réaliser.

[10] crdp.ac-dijon.fr/IMG/pdf_serveur_web.pdf

[11] www.awt.be/contenu/tel/res/res,fr,fic,035,000.pdf

II.5.2. Le réseau de l'entreprise.

Le réseau de l'entreprise comprend l'ensemble des ordinateurs, logiciels et matériels de connectique, généralement reliés en réseau et parfois connecté à l'Internet. La partie sécurisée du réseau d'entreprise utilisée pour partager et/ou diffuser des informations à usage interne à l'aide de l'e-mail, des technologies web, etc, s'appelle l'intranet. Le réseau de l'entreprise peut également contenir des serveurs, à l'usage des internautes, contenant des informations comme:

- l'objectif de l'entreprise (sa carte de visite),
- son catalogue de produits,
- l'e-commerce.
- les bases de données contenant l'état des stocks, etc.

Cette partie est appelée DMZ (zone démilitarisée).

L'entreprise a également la possibilité d'héberger ses sites web d'informations à usage interne ou externe chez un ISP, fournisseur de services Internet.

II.5.3. La communication via l'Internet.

Les applications Internet s'appuient sur le protocole universel TCP/IP (Transmission Control Protocol / Internet Protocol), le type d'accès (modem, adsl, etc.) n'ayant pas d'importance. Cependant, il est important de souligner que si l'on dispose d'une connexion internet temporaire (PSTN ou RNIS), il faut ouvrir une connexion avant de démarrer une communication.

II.5.4. La sécurité.

Internet est un réseau ouvert: tout le monde peut y accéder. Les principes de fonctionnement (l'adressage des ordinateurs, l'aiguillage des connexions dans le réseau, etc.) sont dans le domaine public. De ce fait certaines précautions sont requises pour protéger son réseau et ses informations, éviter le piratage, les intrusions malveillantes, etc.

La sécurité doit être envisagée de manière globale! Il est inutile de dépenser des fortunes pour un système de firewall si le réseau est pénétrable par d'autres accès non protégés (par exemple un PC connecté à l'intranet mais qui dispose également d'un accès individuel à l'Internet via un modem).

A côté de cette sécurisation du réseau de l'entreprise, il convient également de se préoccuper, le cas échéant, de la sécurité au niveau des applications: e-commerce avec paiement en ligne, etc.

II.5.5. La mise en œuvre.

Les étapes de la mise en œuvre sont:

- La définition des besoins,
- La définition de l'architecture,
- La conception (design),
- Le développement, les tests,
- La formation du personnel,
- La mise à disposition.

Viendront ensuite les étapes de maintenance, de mise à jour, etc.

Ces différentes étapes exigent des ressources machines, financières, temps et surtout des ressources humaines. Pour mener à bien un projet, il est nécessaire d'en envisager tous les points sans une seule exception, de travailler avec méthode, de gérer l'avancement des tâches ainsi que l'utilisation des ressources.

Que ce soit pour un usage interne de planification, de communication et de coordination ou pour utiliser de la sous-traitance, le cahier des charges est un outil indispensable pour assurer le succès d'un projet.

Si les ressources humaines ou les compétences requises ne sont pas disponibles, des consultants externes à l'entreprise peuvent apporter l'aide additionnelle nécessaire.

II.5.6. Architecture 3-tiers.[12]

Architecture à 3-tiers ou Architecture à 3 niveaux

Couche présentation : partie de l'application responsable de la présentation des données, et de l'interaction avec l'utilisateur. (Application HTML exploitée par un navigateur Web ou WML pour être utilisée par un téléphone portable par exemple).

Couche métier : reçoit les requêtes utilisateur. Le serveur d'application fournit les traitements métiers. C'est là qu'est implémentée la logique du système et ses règles de gestion. Ce niveau protège les données d'un accès direct par les clients.

[12] http://deptinfo.cnam.fr/new/spip.php?pdoc3392

Couche d'accès aux données : est responsable de la gestion des données. Cette couche permet de rendre l'accès aux données transparentes (uniforme) quelle que soit la méthode utilisée pour les stocker (fichier, base de données...).

Cette architecture est avant tout un élément de structuration logique. Rien n'empêche aux 3 serveurs de s'exécuter sur une même machine.

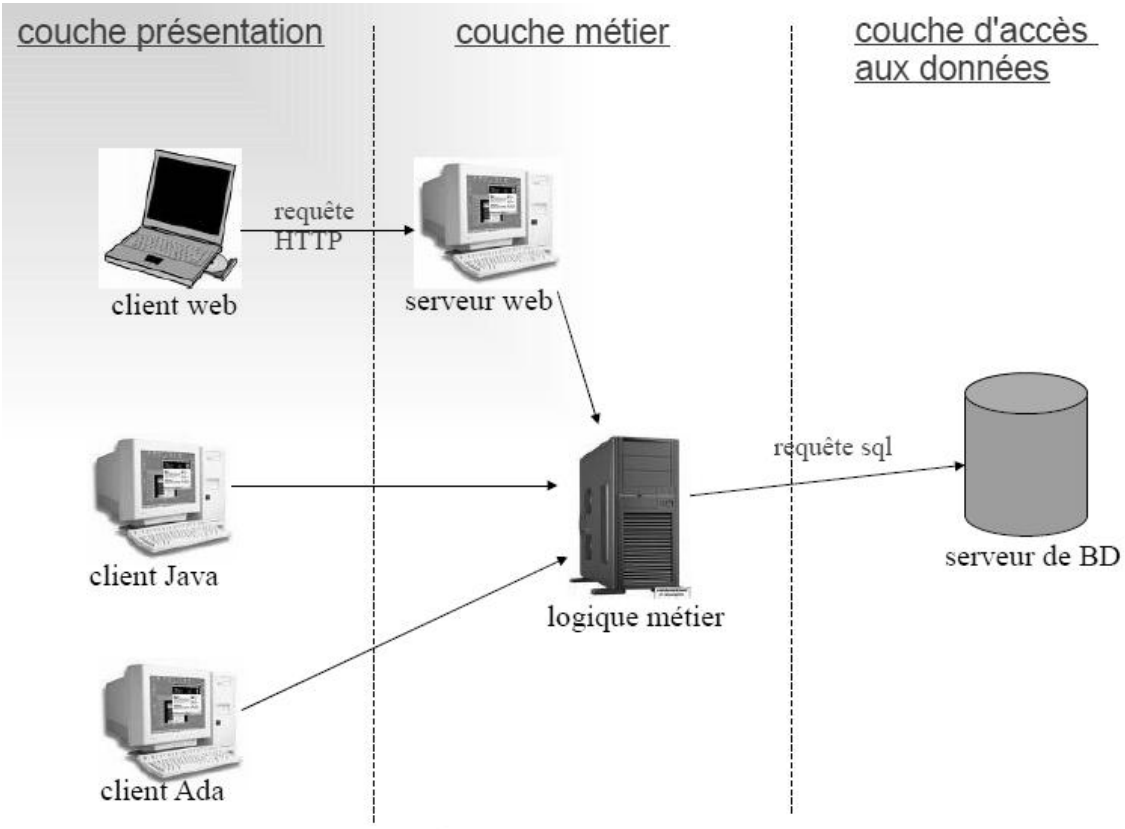


Figure II.5.6. : Architecture à 3-tiers.

II.5.6.1. Avantages (Architecture 3-tiers).

- nette séparation entre les 3 couches favorisant un développement plus rapide d'application Web par réutilisation de composants métiers prédéfinis

- échange de composant facilité.

- protection des données et sécurité rendues plus faciles à obtenir.

- changement d'implémentation de composants possible sans réécriture de l'ensemble de l'application.

II.5.7. J2EE.[13]

J2EE est une plate-forme fortement orientée serveur pour le développement et l'exécution d'applications distribuées et donc en particulier d'applications Web.

L'infrastructure J2EE permet de séparer les applications de l'environnement dans lequel elles s'exécutent Elle est composée de deux parties essentielles :

• une infrastructure de services (transaction, sécurité, ...) dans laquelle s'exécutent les composants écrits en java : un tel environnement se nomme serveur d'application.

• un ensemble d'API (Application Programming Interface) : API Servlet, API JavaServer Pages, API JDBC, ...

Concurrent de .NET

II.5.7.1. Composants J2EE.

II.5.7.1.1. Servlets.

Une servlet est une classe Java dont le rôle est d'étendre les fonctionnalités d'un serveur qui gère des applications accessibles en mode question-réponse. Elle reçoit une question, effectue un traitement et renvoie une réponse.

Communément, les servlets étendent des applications hébergées par un serveur Web. Elle reçoivent donc une requête HTTP, effectue un traitement qui consiste à générer une page HTML et retourne cette page vers le navigateur de l'utilisateur.

La technologie Java Servlet définit des classes spécifiques au mode HTTP.

Les servlets peuvent s'appliquer à d'autres types de requêtes et donc à n'importe quel protocole de type requête/réponse C'est une technologie utilisée par des sites à fort trafic comme ESPN.com ou AltaVista.com.

II.5.7.1.2. JSP.

Java Server Pages est une technologie Java construite au-dessus des servlets.

L'objectif premier est de séparer les parties présentation et génération de contenu d'une application Web.

JSP est un langage de script. Un script JSP est composé d'instructions Java placées entre balises et intégré au code d'une page HTML. Son rôle est de générer du code HTML.

[13] http://deptinfo.cnam.fr/new/spip.php?pdoc3392

JSP n'est pas limité à la génération de texte HTML. Le code JSP permet aussi bien la génération de code XML ou XHTML.

II.5.7.1.3. XML.

XML (eXtensible Markup Language) est un langage : pour l'échange, le stockage et l'affichage de données sur l'Internet

XML est un langage de balises (tags) : il n'utilise pas de balises prédéfinis comme HTML

XML est extensible : il permet de définir de nouvelles balises : c'est un métalangage.

Tout document XML est validé grâce à un autre document (DTD ou XML Schéma) qui contient la grammaire définissant le document XML.

II.6. Conclusion.

Le nombre des technologies, plateformes et cadres d'applications existantes amènes à définir précisément la nature, le public visé, les services rendus par l'application que l'on veut développer, pour choisir les meilleures ressources.

III.1. Introduction.

Avec l'arrivée de l'internet et le Web, beaucoup d'applications ne sont plus développées en utilisant les technologies client/serveur traditionnelles. Au lieu de cela, de nouvelles applications sont développées, en l'occurrence, les applications Web. Une application Web est un système logiciel dont les fonctionnalités sont fournies à travers le Web. Les applications Web existants peuvent avoir besoin de subir à un processus de rétro-ingénierie pour leur maintenance, évolution, migration on compréhension. Pour satisfaire ces besoins, plusieurs approches ont vu le jour.

[Chung and Lee.-2000] proposent une approche de rétro-ingénierie des sites Web prenant le chemin inverse du processus d'ingénierie de logiciel appelé le processus unifié. On obtient comme sortie des diagrammes UML qui peuvent être utilisé pour la maintenance de ce site.

[Filippot and Paolo.-2001] proposent un modèle UML générique pour la représentation du niveau élevé des applications Web. Leur objectif était de tester ces applications.

[Antoniol and al.-2000] rétablissent une vue architecturelle selon les primitives du modèle de données de gestion de relation (RMDM) de la méthodologie de gestion de relation (RMM). Le modèle rétablit est utilisé pour faire la réingénierie des sites Web. Sauf que RMM est mieux adaptée aux sites Web statiques plus qu'aux sites dynamiques.

[Di Lucca and al.-2001] proposent une approche qui définit un ensemble de vues abstraites, modélisées en utilisant des diagrammes UML semblable à celle proposée par J. Conallen [Conallen-1999a ; Conallen1999b], organisés en hiérarchie de différents niveaux d'abstraction, dépeignant plusieurs aspects d'une application Web pour faciliter sa compréhension.

[Hassan and Richard.-2002] proposent une approche pour récupérer l'architecture des applications Web, afin de rendre la maintenance plus maniable. Certes que c'est une approche riche par rapport aux autres approches en terme de source d'information à partir de laquelle on extrait l'architecture, car elle traite les scripts de programmation aussi bien que les balises HTML.

Mais nous croyons que le schéma résultant n'est pas riche car il ne contient que des boites et des graphes simples.

Ces approches et d'autres n'ont pas tenu en compte que les applications Web à partir desquelles on extrait des schémas conceptuels peuvent contenir des erreurs commises par les concepteurs originaux de ces applications. En plus de ça elles s'intéressent beaucoup plus à l'architecture globale du système et non pas au schéma de la base de données.

[Fabrice and al.-2003] proposent une approche méthodologique pour la rétro-ingénierie des pages Web, supporté par des outils. Ces résultats sont la base nécessaire de la ré-documentation de site Web et la migration de données vers un nouveau système d'information.

[Fabrice and al.-2003] ont essayé de couvrir la pauvreté sémantique des documents HTML par un enrichissement sémantique orienté utilisateur, mais cela réduit l'automatisation du processus de rétro-ingénierie des applications Web. En plus de ça cette approche ne s'applique que sur des sites Web relativement bien structurés.

Pour couvrir la pauvreté sémantique des documents HTML sans réduire l'automatisation du processus de rétro-ingénierie, on propose dans ce mémoire une approche de rétro-ingénierie des applications Web à base d'ontologie de domaine pour générer un schéma conceptuel modélisant l'application Web.

III.2. Approche de rétro-ingénierie des applications Web à base d'ontologie.

Initialement, on suppose que :

a) Dans des pages HTML, un champ d'un tableau ou un élément d'une liste peut désigner un concept d'une ontologie de domaine ; ou bien l'ensemble des champs d'un tableau ou l'ensemble des éléments d'une liste peut désigner un concept d'une ontologie de domaine.

Par exemple, soit la page HTML :

```
<html>
…
 <table>
  <tr> <td>Code</td> <td>Nom</td>          <td>Société</td> </tr>
  <tr>     <td> 1 </td>       <td>Ahmed</td> <td>ENIEM/1</td> </tr>
 </table>
…
 <ul>      <li>Désignation : Tomate</li>       <li>Quantité : 150</li>     </ul>
</html>
```

Cette page HTML contient un tableau possédant les champs "Code", "Nom" et "Société". Le champ "Société" est équivalent au Concept 'Société" de l'ontologie de domaine du "Personnel". L'ensemble de champs {"Code", "Nom"} est équivalent au concept "Employé" de la même ontologie. Cette page HTML, contient aussi une liste possédant les éléments {"Désignation", "Quantité"}…

b) II existe une ontologie, construite par des experts et elle est spécifique à un domaine (domaine auquel appartiennent les pages HTML). Cette ontologie contient bien sûr des concepts, des relations...

c) A partir d'une ontologie de domaine, on peut extraire le schéma conceptuel global décrivant le domaine tout entier.

L'objectif de notre approche est d'extraire un sous schéma riche et réduit décrivant l'application Web à base de l'ontologie de domaine.

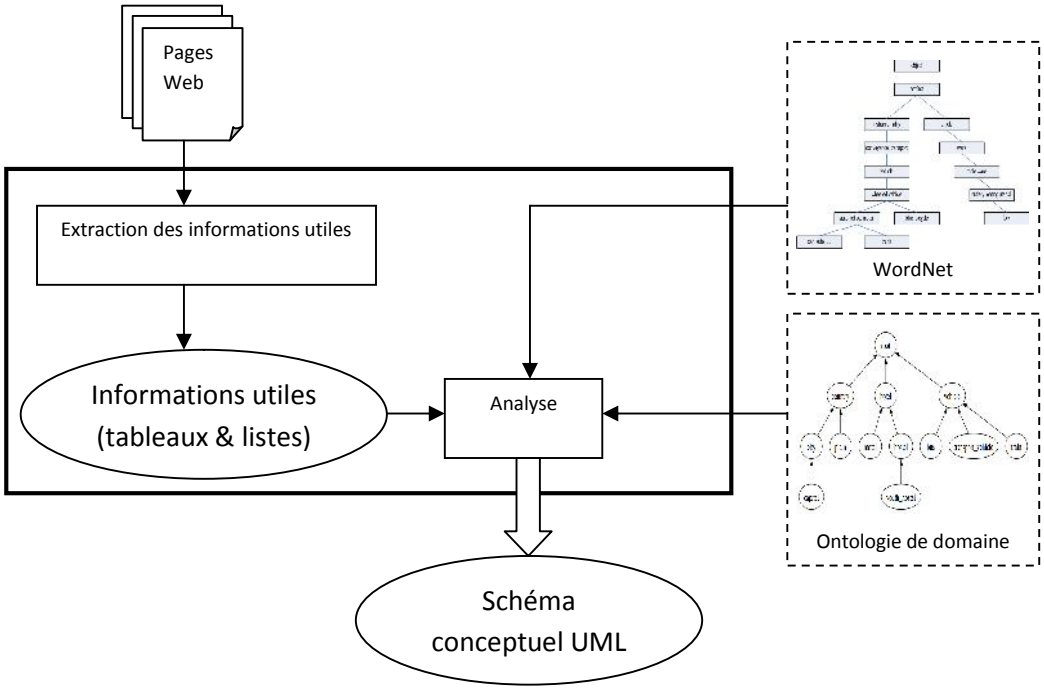


Figure III.2. : Processus de Rétro-ingénierie des applications Web à base d'ontologie.

Le processus de rétro-ingénierie consiste en deux grandes phases : Tout d'abord, On doit extraire les informations utiles à partir des pages HTML pour les comparer par rapport aux informations présentées dans l'ontologie ; Dans notre cas, les informations utiles sont celles présentées sous forme de tableaux et de listes, car c'est la forme la plus utilisée pour présenter des informations bien structurées dans une page Web. La deuxième étape est l'Analyse qui consiste à identifier ou à reconnaître les concepts de l'ontologie cachés dans les pages Web à l'aide des techniques de distance sémantique, puis inférer de nouveaux concepts et relations, et enfin générer un schéma conceptuel UML (figure III.2).

Notre approche peut être utilisée pour la rétro-ingénierie des applications Web incluant des pages dynamiques avec un contenu qui se change continuellement. Elle est beaucoup plus orienté données, c.-à-d., qu'elle décrit l'aspect statique de l'application Web (Schéma de la base de données).

Dans ce qui suit, on présente en détail les phases du processus de rétro-ingénierie des applications Web à base d'ontologie.

III.2.1. Extraction des informations utiles.

Cette phase commence par le filtrage des documents HTML, suivi de l'extraction du DOM[1] et en fin l'extraction des informations utiles à partir du DOM (figure III.2.1). Le filtrage consiste à parcourir le code source des pages HTML, éliminer les balises inutiles tel que celles de la mise en forme (par exemple <b>, <i>,...), et conserver les balises utiles, qui portent les informations à traiter dans les étapes suivantes (par exemple <html>, <body>, <table>, <td>, <tr>, <ul>, <li>,...). Le résultat de cette étape est un ensemble de pages HTML nettoyées.

Lors de L'extraction des informations utiles, plusieurs problèmes peuvent se révéler tels que :

- Tableaux et listes de présentation plutôt que d'information.
- Tableaux et listes imbriqués.
- Positions des champs (horizontal ou vertical).

Pour traiter ces problèmes et d'autres. On procède à plusieurs techniques telles que celles de compréhension des tableaux.

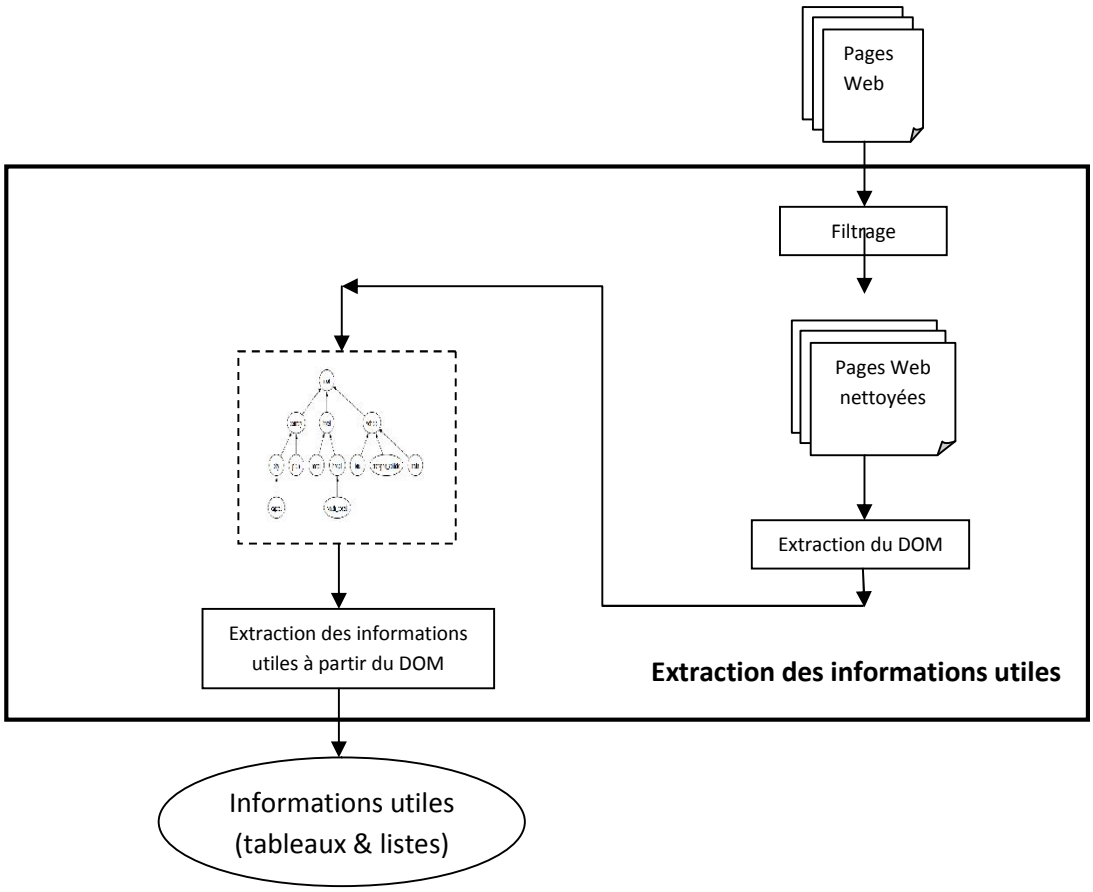


Figure III.2.1. : La phase d'Extraction des informations utiles.

[1] Le DOM un Document Object Model est une API qui consiste à décomposer le contenu d'un document HTML ou XML en une arborescence de nœuds (chaque élément du document est un nœud).

Les pages HTML nettoyées seront présentées en format logique DOM pour pouvoir les manipuler. A partir de ce format logique DOM, on peut extraire maintenant les informations utiles, stockées dans les tableaux et les listes.

III.2.2. Analyse.

La phase d'analyse est une suite d'étapes pour le traitement des informations utiles issues de la phase précédente. Le résultat de cette phase est un schéma conceptuel UML (figure III.2.2.1).

III.2.2.1. Analyse morphologique.

La phase d'analyse débute par une analyse morphologique appliquée sur les champs des tableaux et les éléments des listes extraits à partir des pages HTML. L'analyse morphologique consiste par exemple à enlever les traits d'union et garder uniquement les racines des termes tel quels apparaissent dans WordNet[2].

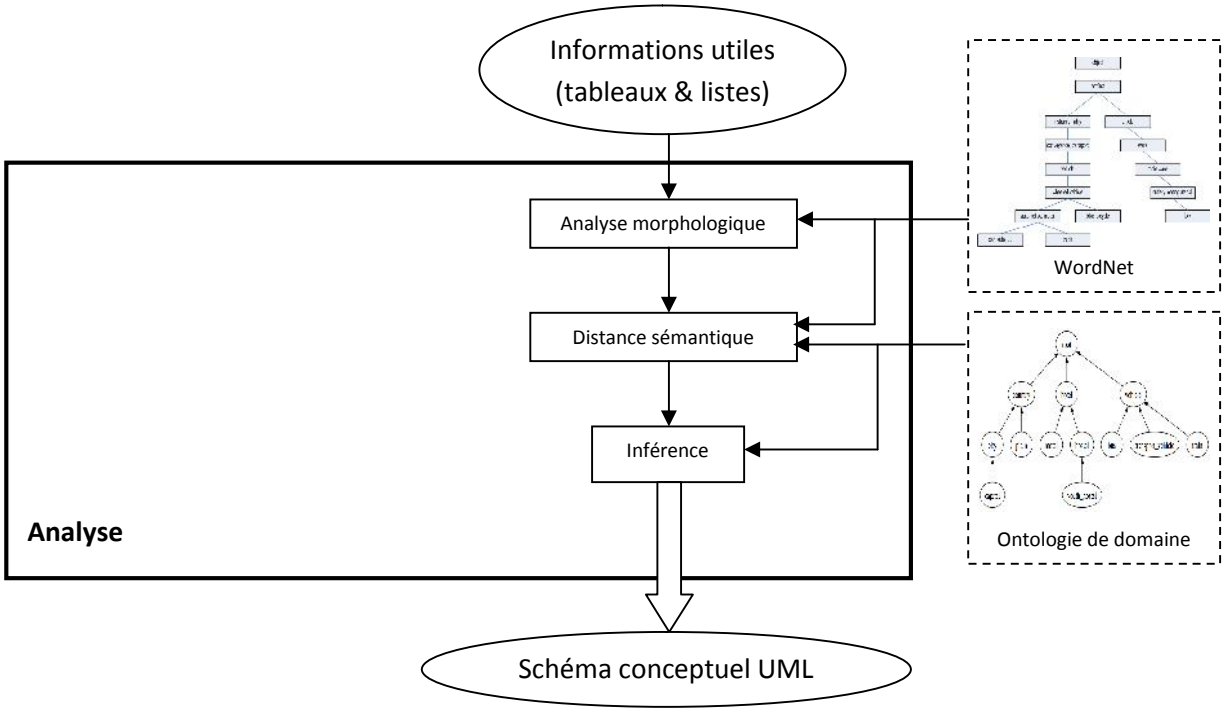


Figure III.2.2.1. : La phase d'analyse.

III.2.2.2. Distance sémantique.

Pour mesurer la similarité, ou encore pour calculer la distance sémantique entre deux concepts (Cette distance est appelée dans ce mémoire la distance sémantique élémentaire), plusieurs approches ont étaient proposées. Elles visent à quantifier le degré de ressemblance sémantique entre deux concepts. Certaines de ces approches se basent sur WordNet. La mesure LCH [Leacock

[2] WordNet est une base de données lexicale qui organise les noms et les verbes dans des concepts *(synset)* en hiérarchies de relations *is-a.* Chaque concept est décrit par une brève glose (Referez vous à l'annexe C sur WordNet).

and al.-1998] recherche le plus court chemin entre deux concepts et multiplie cette valeur par le maximum des deux chemins là ou ils apparaissent dans l'hiérarchie is-a.

WUP [Wu and al.-1994] recherche la longueur du chemin vers la racine à partir du plus proche père commun (PPC) des deux concepts, qui est le concept le plus spécifique dont ils partagent comme ancêtre. Cette valeur est multipliée par la somme des deux longueurs des chemins à partir de chaque concept vers la racine. La mesure path est égale à l'inverse de la longueur du plus court chemin entre les deux concepts. La mesure RES [Resnik and al.-1995] est basée sur le contenu d'information. Les mesures LIN [Lin and al.-1998] et JCN [Jiang and a1.-1997] augmentent le contenu d'information du PPC des deux concepts avec la somme des contenus d'information de chaque concept. La mesure LIN multiplie le contenu d'information du PPC par cette somme, tant que JCN fait la soustraction du contenu d'information du PPC de cette somme (puis elle prend l'inverse pour le convertir d'une distance vers une mesure de similarité). La mesure HSO [Hirst and al.-1998] est basée sur le chemin, et classifie les relations dans WordNet selon leurs directions. Elle évalue la similarité entre les concepts en asseyant de rechercher le plus court chemin avec le moindre changement de direction. La mesure LESK [Banerjee and al.-2003] calcule la similarité en recherchant et enregistrant le chevauchement entre les gloses des deux concepts. Aussi bien que les concepts ayant un lien direct avec eux selon WordNet. Pour la mesure VECTOR [Patwardhan and al.-2003], chaque terme utilisé dans une glose WordNet est associé à un vecteur de contexte. Chaque glose est représentée par un vecteur de glose qui est la moyenne de tous les vecteurs de contexte des termes trouvés dans la glose. La similarité entre les deux concepts est égale au cosinus entre les deux vecteurs de glose.

Maintenant pour calculer la distance sémantique entre deux groupes d'entités, plusieurs stratégies sont utilisées. Certaines de ces stratégies sont utilisées pour le groupement hiérarchique [Alexander-2003].

Single linkage : La distance entre deux groupes est celle des deux membres les plus proches (Similaires) dans ces deux groupes. Elle est calculée comme suit :

D(r,s) = Min { d(i,j) : ou i est dans le groupe r et j est dans le groupe s }

Complete linkage : La distance entre deux groupes est celle des deux membres les plus dissimilaires dans ces deux groupes. Elle est calculée comme suit :

D(r,s) = Max {d(i,j) : ou i est dans le groupe r et j est dans le groupe s}

Average linkage : La distance entre deux groupes est la moyenne des distances entre les membres des deux groupes.

$D(r,s) = T_{rs} /(N_r * N_s)$; ou T_{rs} est la somme de toutes les distances, N_r et N_s est le nombre d'éléments de r et de s.

Une autre stratégie utilisée pour le calcul de la similarité entre ensembles dans des ontologies [Marc and al.-2004].

Multidimensional scaling : C'est une technique statistique pour le calcul de la distance sémantique entre deux ensembles d'entité. Pour cela, on décrit chaque entité par un vecteur représentant la similarité à chacune des entités de l'autre ensemble. Pour les deux ensembles un vecteur représentatif peut être créé par le calcul de la moyenne des vecteurs de tous les individus. Finalement. On détermine le cosinus entre les deux vecteurs d'ensemble par le produit scalaire pour obtenir une valeur de similarité. La formule de calcul est la suivante :

$$sim_{set}(E,F) := \frac{\sum_{e\in E}\vec{e}}{\left|\sum_{e\in E}\vec{e}\right|} \cdot \frac{\sum_{f\in F}\vec{f}}{\left|\sum_{f\in F}\vec{f}\right|}, \text{Avec l'ensemble d'entités } E = \{e_1, e_2, \ldots\},$$

$$\vec{e} = (sim(e,e_1), sim(e,e_2), \ldots sim(e,f_1), sim(e,f_2), \ldots); F \text{ et } f \text{ sont définis analogiquement}$$

La première étape de multidimensional scaling est de transformer les distances (ou les similarités) en cordonnées absolues. Les coordonnées sont essentiellement les similarités entre un couple d'objets. Pour leur objectif, [Marc and al.-2004] ont étendu cela et ont créé la mesure de similarité entre ensembles présentée ici. Ils construisent la moyenne des coordonnées de l'ensemble en faisant la sommation et la normalisation. Enfin, ils calculent la distance-cosinus entre les deux vecteurs (coordonnées) des deux ensembles. Ils notent que multidimensional scaling ne transforme pas uniquement les similarités en coordonnés absolues, mais elle réduit aussi les dimensions à deux par exemple, pour présenter les similarités dans un diagramme. Ils affirment aussi que leur approche est beaucoup plus convenable si la distance est prise entre 0 et 1, comme c'est le cas dans ce mémoire. Pour comprendre cette stratégie, prenant l'exemple suivant :

Soit deux ensemble A= {a1, a2, a3} et B={b1, b2}, on désire calculer la distance entre les deux ensembles sachant que les distances entre chaque deux objets (distance sémantique élémentaire) sont comme suit :

sim(a1,b1)=0.22 , sim(a1,b2)=0.19 , sim(a2,b1)=0.99 , sim(a2,b2)=0.00 , sim(a3,b1)=0.66 ,

sim(a3,b2)=0.23 , sim(a1,a1)=1 , sim(a1,a2)=0 , sim(a1,a3)=0 etc.

Comme ça on peut remplir la matrice suivante :

	a1	a2	a3	b1	b2
al	1	0	0	0.22	0.19
a2	0	1	0	0.99	0
a3	0	0	1	0.66	0.23
b1	0.22	0.99	0.66	1	0
b2	0.19	0	0.23	0	1

e_a1 = (sim(a1,a1), sim(a1,a2), sim(a1,a3), sim(a1,b1), sim(a1,b2)) = (1, 0, 0, 0.22, 0.19)
e_a2 = (0, 1, 0, 0.99, 0)
e_a3 = (0, 0, 1, 0.66, 0.23)
sum_E/A = (1, 1, 1, 1.87, 0.42)

f_b1 = (0.22, 0.99, 0.66, 1, 0) f_b2 = (0.19, 0, 0.23, 0, 1)
sum_F/B = (0.41, 0.99, 0.89, 1, 1)

sim_set(E,F) = sum_E/A * sum_F/B / (|sum_F/A|*|sum_F/B|)= 4.58 / 5.11 = 0.895

Revenons à notre approche. On a d'un coté, chaque tableau ou liste correspond à une entité anonyme, décrite par ses champs ou ses éléments. De l'autre coté, une ontologie de domaine contenant des concepts. Chaque concept possède éventuellement un ensemble d'attributs. Maintenant il va y avoir des calculs de distance sémantique pour identifier les concepts de l'ontologie qui existent dans les pages HTML.

Pour cela on doit fixer un seuil pour la distance sémantique (distance élémentaire ou distance entre groupe). Le seuil est une valeur entre 0 et 1, la valeur 1 indique que les deux concepts sont équivalents à 100%. Puis on doit choisir une méthode pour le calcul de la distance sémantique élémentaire et une stratégie pour le calcul de la distance sémantique entre deux groupes d'entités.

Si on trouve que la distance sémantique élémentaire entre un champ d'un tableau (ou bien un élément d'une liste) et le nom d'un concept de l'ontologie est supérieure au seuil fixé auparavant alors on considère que le concept de l'ontologie est identifié ou bien reconnu ou tout simplement existant.

Sinon, si on trouve que la distance sémantique entre deux groupes : le premier groupe est l'ensemble des champs du tableau (ou bien l'ensemble des éléments de la liste), le deuxième groupe est l'ensemble des attributs du concept de l'ontologie ; si on trouve que cette distance est supérieure au seuil alors on considère que le concept de l'ontologie est identifié.

L'algorithme de calcul de distance sémantique est le suivant :

- Charger les pages HTML.
- Charger l'ontologie de domaine exprimée en OWL-DL.
- Fixer un seuil pour la distance sémantique. choisir une méthode pour le calcul de distance sémantique élémentaire et choisir une stratégie de calcul de distance sémantique entre groupes.
- Chaque tableau correspond à un objet anonyme ayant comme attributs les champs de ce tableau ; Aussi, chaque liste correspond à un objet anonyme ayant comme attributs les éléments de cette liste ; Tous ces objets anonymes sont stockés dans un vecteur d'objet V1.
- Chaque concept de l'ontologie correspond à un objet ayant comme nom le nom du concept et comme attributs les attributs de ce concept. Tous ces objets nommés sont stockés dans un vecteur de concept V2.
- ***Pour chaque*** objet O_i du vecteur V1 ***Faire***
    - ***Pour chaque*** objet O_j du vecteur de concepts V2 ***Faire***
        - Calculer la distance sémantique entre le nom de l'objet O_j et les attributs de l'objet O_i un par un ;
        - ***Si*** pour un attribut de O_i la distance est supérieure au seuil, ***Alors*** marquer l'objet O_j comme objet existant.
        - ***Sinon***

- calculer la distance sémantique entre les deux groupes en utilisant la stratégie choisie ; le premier groupe est l'ensemble des attributs de l'objet O_i, le deuxième est l'ensemble des attributs de l'objet O_j.
- ***Si*** la distance entre les deux groupes est supérieure au seuil ***Alors*** marqué l'objet O_j comme objet existant.
- ***Fin Si***

- ***Fin Si***

- ***Fin Pour***

- ***Fin Pour***

Complexité : La fonction de complexité de cet algorithme est : $O(n^2 \ C_{sim})$, avec C_{sim} est la fonction de complexité de la l'algorithme de calcul de distance sémantique qui dépend de l'approche choisie. Si $C_{sim} = O(n^k)$ alors la complexité de cet algorithme sera $O(n^k)$ implique que cet algorithme sera polynomial. Si $C_{sim} = O(c^n)$ alors la complexité de cet algorithme sera $O(c^n)$ implique que cet algorithme sera non polynomial.

III.2.3. Inférence.

L'inférence consiste à inférer de nouvelles relations et de nouveaux concepts, avant de générer le schéma conceptuel UML, décrivant l'application Web à base d'ontologie (figure III.2.3).

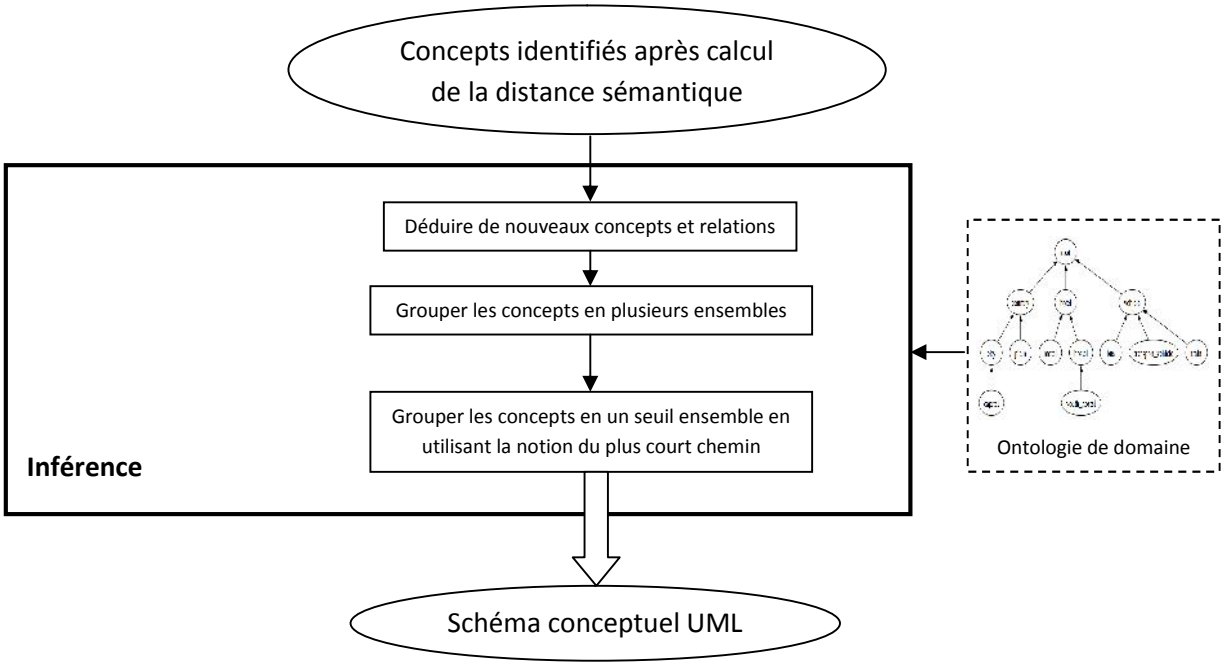


Figure III.2.3. : Le processus d'inférence.

L'inférence commence par la déduction de nouveaux concepts et relations. A partir des concepts identifiés dans l'étape précédente, on peut enrichir notre schéma conceptuel cible en déterminant les concepts en relation avec les premiers concepts identifiés auparavant, et cela en utilisant bien sur l'ontologie ; ou autrement dit, chaque relation possède d'un coté un concept identifié, elle doit apparaître dans le schéma conceptuel ainsi que tous les concepts de ses deux cotés. L'objectif de cette étape est l'enrichissement du schéma conceptuel résultant. ***L'algorithme de déduction*** est le suivant :

- Chaque relation de l'ontologie correspond à un objet ayant comme nom le nom de la relation, et comme attributs deux vecteurs VD et VR, indiquant les concepts du *Domaine* et du *Range* de la relation. Tous ces objets seront stockés dans on vecteur de relation V3.
- ***Répéter***
    - ***Pour cheque*** objet O_i du vecteur de concept V2 ***Faire***
        - ***Pour cheque*** objet O_j du vecteur de relation V3 ***Faire***
            - ***Si*** O_i est marqué comme objet existant ***Alors***
                - ***Si*** O_i apparait dans *Domaine* ou *Range* de O_j ***Alors*** marquer O_j comme objet existant.
                - ***Fin Si***
            - ***Fin Si***
        - ***Fin Pour***
    - ***Fin Pour***
    - ***Pour cheque*** objet O_j du vecteur de relation V3 ***Faire***
        - ***Si*** O_j est marqué comme objet existant ***Alors***
            - ***Pour cheque*** objet O_d du vecteur du *Domaine* VD ***Faire***
                - ***Si*** l'objet équivalent à O_d dans le vecteur des concepts V2 n'est pas marqué comme objet existant ***Alors*** le marqué comme objet existant.
                - ***Fin Si***
            - ***Fin Pour***
            - ***Pour Chaque*** objet O_r du vecteur du *Range* VR ***Faire***
                - ***Si*** l'objet équivalent à O_r dans le vecteur des concepts V2 n'est pas

marqué comme objet existant ***Alors*** le marqué comme objet existant.

- ***Fin Si***
- ***Fin Pour***

- ***Fin Si***

- ***Fin Pour***

- ***Jusqu'à*** ce qu'aucun nouvel objet n'est marqué comme objet existant dans le vecteur des concepts V2.

Complexité : La fonction de complexité de cet algorithme est : $O(n^4)$, implique que cet algorithme est polynomial.

Maintenant on va obtenir un ensemble de concepts et de relations avec lesquels on peut former un ensemble de groupes. Chaque groupe représente un graphe connexe[3]. ***L'algorithme de groupement des concepts en plusieurs groupes*** est le suivant :

- Créer on vecteur d'ensemble V4 contenant des objets. Chaque objet a un attribut décrivant la liste des éléments de l'ensemble correspondant.
- ***Pour chaque*** objet O_j du vecteur de relation V3 ***Faire***
  - ***Si*** 0_j est marqué comme objet existant ***Alors***
    - Les objets du vecteur de *Domaine* VD et du vecteur de *Range* VR vont construire les éléments d'un nouveau ensemble ajouté comme objet dans le vecteur d'ensemble V4.
  - ***Fin Si***
- ***Fin Pour***
- ***Répéter***
  - ***Pour chaque*** objet 0_e du vecteur d'ensemble V4 ***Faire***
    - ***Pour chaque*** objet O_k du reste du vecteur d'ensemble V4 ***Faire***
      - ***S***'il y a un élément commun entre la liste des éléments de l'objet O_e et celle de l'objet O_k ***Alors***

[3] Un graphe est connexe si et seulement s'il existe un chemin entre n'importe quelle paire de sommets distincts du graphe.

- Calculer l'union des deux listes.
- Affecter l'union des deux listes à l'objet O_e.
- Supprimer l'objet O_k du vecteur d'ensemble V4.

- ***Fin Si***

- ***Fin Pour***

- ***Fin Pour***

- ***jusqu'à*** ce qu'il n'y a plus de modification dans le vecteur d'ensemble V4.

Complexité : La fonction de complexité de cet algorithme est : $O(n^4)$, implique que cet algorithme est polynomial.

Avant de générer le schéma conceptuel final, les groupes issus de l'étape précédente doivent être unifiés en un groupe unique, sans pour autant encombrer le schéma par plusieurs autres concepts et relations. Chaque deux groupe peuvent être unifiés par le plus court chemin qui peut exister entre un concept du premier groupe et un autre concept du deuxième groupe dans l'hiérarchie de l'ontologie. ***L'algorithme du groupement de ces concepts en un seul ensemble*** est le suivant :

- ***Répéter***
  - ***Pour chaque*** objet O_e du vecteur d'ensemble V4 ***Faire***
    - ***Pour chaque*** objet O_k du vecteur d'ensemble V4 ***Faire***
      - Appeler l'algorithme de plus court chemin pour calculer le plus court chemin qui peut exister entre deux éléments des deux listes des objets O_e, O_k.
    - ***Fin Pour***
  - ***Fin Pour***
  - Ajouter les éléments du plus court chemin au vecteur de relation V3.
  - Appeler l'algorithme de groupement de concepts en plusieurs ensembles.
- ***Jusqu'à*** obtenir un seul ensemble.
- Marquer les nouveaux objets qui apparaissent dans les plus courts chemins comme objets existants dans le vecteur V2.

Complexité : La fonction de complexité de cet algorithme est : $O(n\ (n^2\ C_{pcc}+C_{gcp}))$, avec C_{pcc} est la fonction de complexité de l'algorithme du plus court chemin et C_{gcp} est la fonction de complexité de l'algorithme de groupement des concepts en plusieurs groupes. Puisqu'on a $C_{pcc} = O(n^2)$ et $C_{gcp} = O(n^4)$ alors la complexité de cet algorithme est : $O(n^5)$ implique que cet algorithme est polynomial.

La relation de subsumption n'est pas tenue en compte lors de l'étape de déduction de nouveaux concepts et relations, c.-à-d. que si un concept est identifié ou reconnu, cela ne veut pas dire que son père ou son fils existe. La relation de subsomption apparait dans les plus courts chemins lors de l'unification des groupes en un seul ensemble. La relation de subsomption sera traduite par un lien d'héritage dans le schéma conceptuel. ***L'algorithme de recherche du plus court chemin*** est le suivant :

- Pour un concept C1, chercher tous les chemins possibles accédant à C1 a partir du TOP, et les stocker dans un vecteur *path1*.
- Pour un concept C2, chercher tous les chemins possibles accédant à C2 a partir du TOP, et les stocker dans un vecteur *path2*.
- ***Pour chaque*** élément du vecteur *path1* ***Faire***
    - ***Pour chaque*** élément du vecteur *pcath2* ***Faire***
        - Extraire les chemins avant la même racine et les stocker dans un vecteur *path*.
    - ***Fin Pour***
- ***Fin Pour***
- Choisir le plus court chemin à partir du *path*.

Complexité : La fonction de complexité de cet algorithme est : $O(n^2)$, implique que cet algorithme est polynomial.

III.2.4. Conceptualisation.

Du enrichi de concepts et de relations extraites de la phase précédente, nous pouvons construire un schéma conceptuel UML (Fig. III.2.4). Comme suit: chaque concept et de la relation de l'ensemble extraits seront présentés respectivement par une classe UML et de la relation dans la suite schéma. La relation exprimée par l'expression «partie-de» sera présenté comme une relation d'agrégation UML.

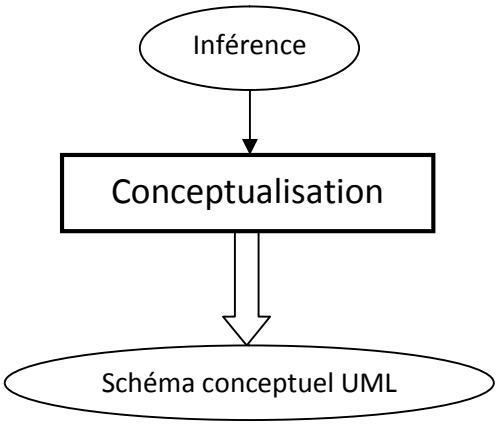


Figure III.2.4. : Phase de conceptualisation.

Les relations de subsomption qui figurent dans les plus courts chemins lors de l'unification des groupes dans un seul jeu seront traduites par un lien de patrimoine dans le schéma conceptuel. Même l'héritage multiple peut apparaît dans ce schéma.

Cardinalités des relations sont également extraites de l'ontologie de domaine qui sera présenté dans le schéma conceptuel UML.

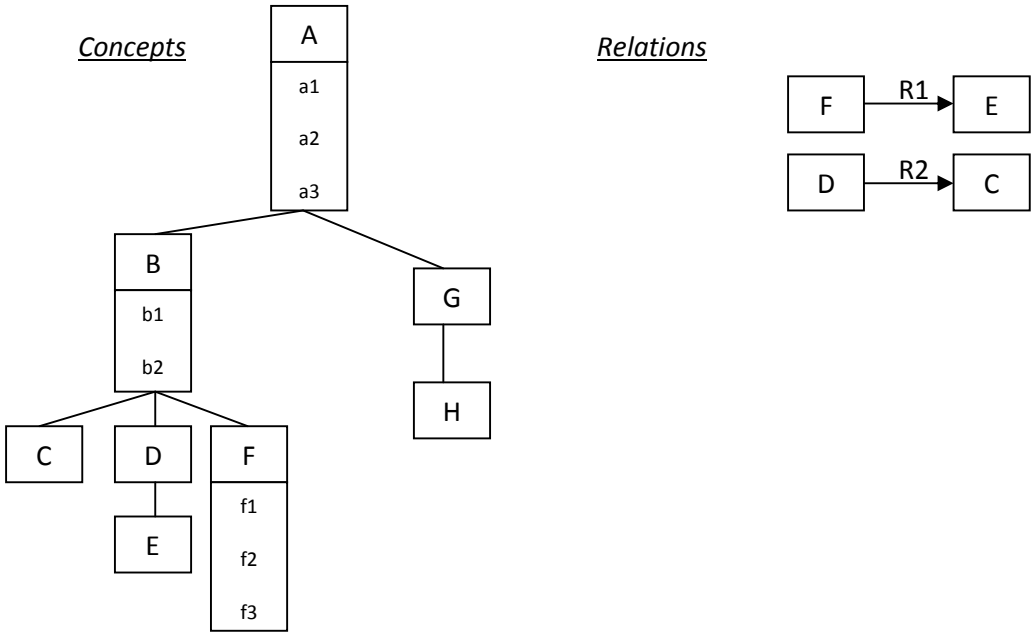
III.3. Exemple prototype.

Pour mieux comprendre l'approche, on va dérouler le processus de retro-ingénierie des applications Web sur un exemple prototype non réel.

Nous supposons que les pages HTML en entrée à notre système contiennent les informations suivantes :

- Un tableau possède les champs t1, t2.
- Une liste possède les éléments m1, m2, m3.

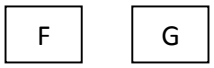
Et on suppose aussi qu'on a l'ontologie de domaine suivante :



A près avoir fixé un seuil et choisir une méthode (par exemple Seuil = 0.7 & comme méthode on choisi *Multidimensional scaling*) pour le calcul de la distance sémantique entre les informations des pages HTML et celles de l'ontologie, on suppose qu'on a obtenu les résultats suivant :

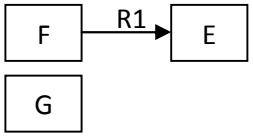
- Distance sémantique élémentaire entre le champ t2 du tableau et le concept G est égale à 0.9 > Seuil et par conséquent le concept G est marqué comme existant.
- Distance sémantique entre le groupe {m1, m2, m3} (i.e. les éléments de la liste) et les attributs {f1, f2, f3} du concept F est 0.85 > Seuil et par conséquent le concept F est marqué existant.
- Les autres distances (distances élémentaires ou entre groupes) sont toutes < Seuil. Par conséquent, les concepts concernés ne sont pas intéressants pour le moment.

Apres cette première étape, notre schéma conceptuel vient de se naitre, mais il ne contient pour le moment que deux concepts.



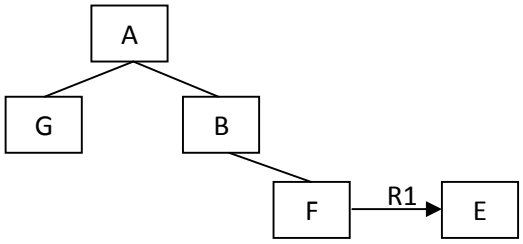
1- On commence l'inférence par *la déduction de nouveaux concepts et relation*. A partir du concept F, on a déduire la relation R1, et comme cette dernière est considérée comme relation passante de la propriété d'existence alors le concept E est marqué aussi comme existant. A partir du concept G on ne peut rien déduire.

Le schéma conceptuel s'évolue maintenant, et contient deux ensemble disjoints : {F, E} et {G}.



2- Pour avoir un graphe connexe, le processus d'inférence se poursuit et les deux *ensembles doivent être groupés* par le plus court chemin entre ces deux ensembles.

Le schéma conceptuel final est représenté comme suit :



III.4. Conclusion.

Dans ce chapitre, on vient de donner une description détaillée de notre contribution dans le domaine de rétro-ingénierie des applications Web. On a comencé par décrire notre approche basée sur l'ontologie qui consiste en deux grandes phases : extraire les informations utiles à partir des pages HTML pour les comparer par rapport aux informations présentées dans l'ontologie. La deuxième étape est l'*Analyse* qui consiste à identifier ou à reconnaitre les concepts de l'ontologie cachés dans les pages Web à l'aide des techniques de distance sémantique, puis inférer de nouveaux concepts et relations, et enfin générer un schéma conceptuel UML. On a essayé aussi de décrire les algorithmes utiliser pour la mise en œuvre des étapes du processus de rétro-ingénierie. On a cloturé ce chapitre par un exemple prototype expliquant en détail les phases de notre approche.

IV.1. Expérimentation.

IV.1.1. Introduction.

Supportant l'approche proposée dans ce mémoire, on a développé un outil qu'on a nommé RetroWebOnto pour dire la rétro-ingénierie des applications orientées web à base d'ontologie. Le système reçoit comme entrées des pages HTML bien formés[1] + ontologie exprimée en OWL-DL. Le système est une implémentation des algorithmes présentés dans le chapitre précédent pour générer à la fin une matrice qui contiennent tous les concepts issus des pages HTML avec l'information « valide ou non valide » par rapport à l'ontologie en utilisant comme outil sémantique Wordnet.

Pour l'évaluation de l'approche de rétro-ingénierie des applications web proposée dans ce mémoire, on va présenter dans ce chapitre une étude de cas. Pour cela nous avons utilisé une ontologie préexistante et un site web sur lequel on va faire l'expérimentation. Dans la fin de ce chapitre, on va discuter sur les résultats obtenus.

IV.1.2. L'ontologie du domaine de tourisme.

L'ontologie qu'on a choisie est une ontologie tutorial pour le web sémantique[2]. Elle décrit le domaine de tourisme. Elle est proposée par Mr Holger Knublauch[3].

[1] Un document HTML bien formé est celui qui possède pour chaque balise ouvrante une balise fermente. Un document no bien formé ne sera pas tenu en compte.
[2] http://protege.stanford.edu/plugins/owl/owl-library/travel.owl
[3] http://www.Knublauch.com/

IV.1.3. Présentation du site Web.

Le site web qu'on a choisi pour faire notre expérimentation est http://www.usatourist.com/, un site web touristique pour les états unis d'Amérique.

Comme entrées à notre système, on a choisi deux pages HTML. Le premier est celle qui présente les états d'USA, la deuxième est celle qui présente les hôtels.

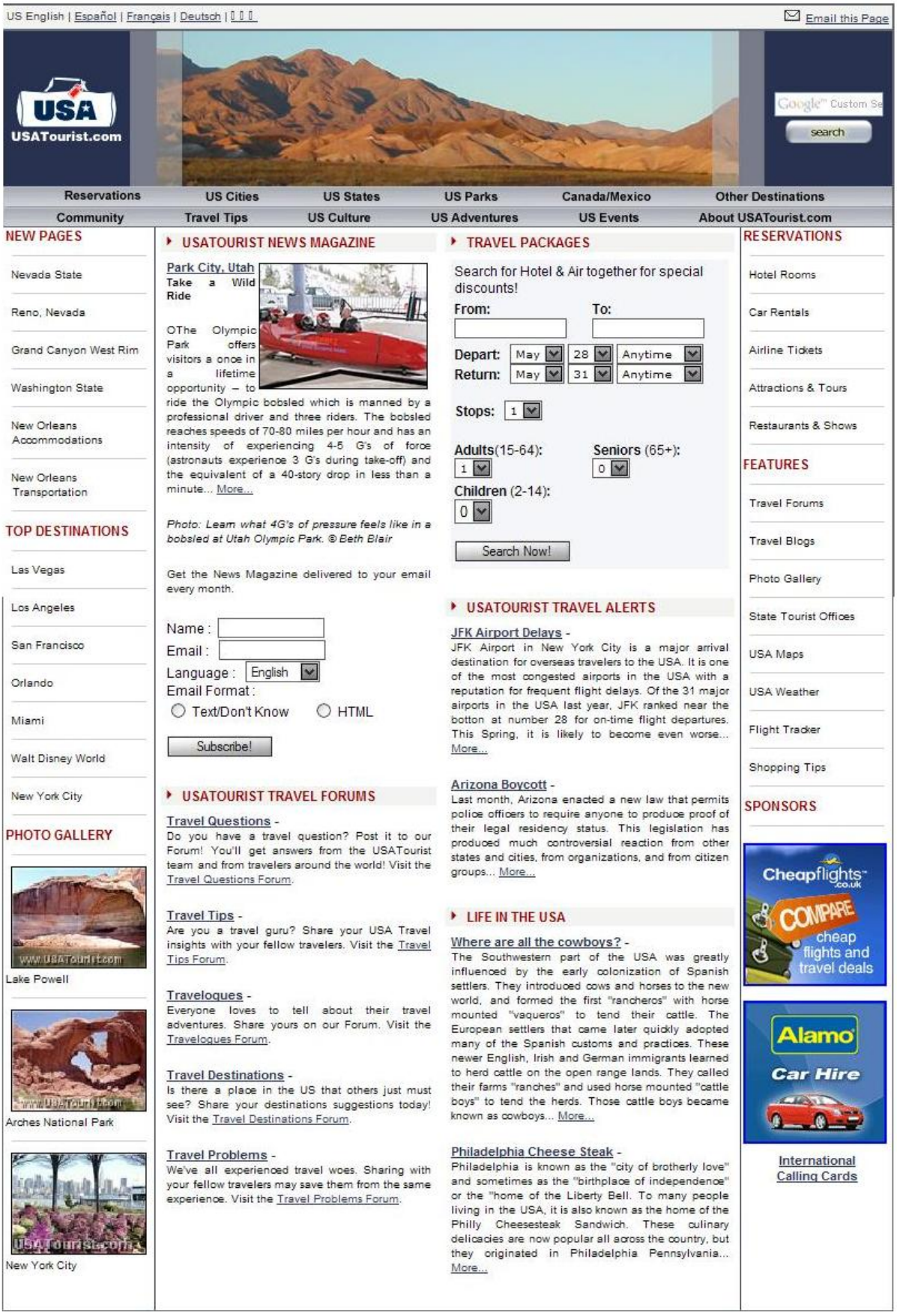


Figure IV.1.3.a : Page d'accueil du site.

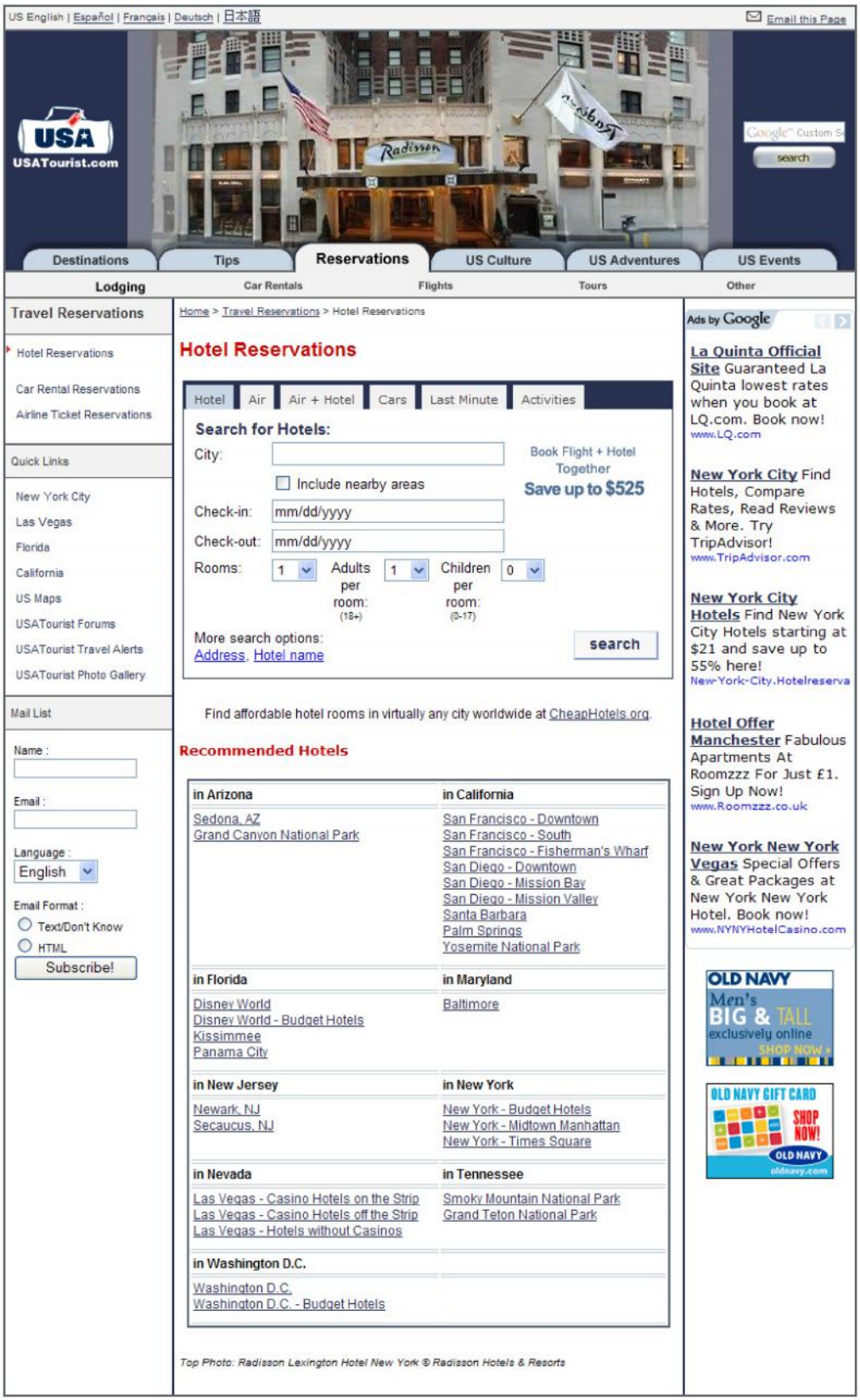


Figure IV.1.3.b : Page des hôtels.

IV.2. Implémentation.

IV.2.1. Introduction.

Après avoir expliqué dans les chapitres précédents, l'aspect théorique de toutes les notions de notre projet, il nous serait intéressant de franchir l'aspect pratique par une implémentation de méthode abordé dans le chapitre 3 pour La Rétro-ingénierie des applications orientée web à base d'ontologie.

Le but de notre projet est de réaliser une application permettant :

1. L'extraction des termes à partir des pages HTML (citées dans les figures IV.1.3.a et IV.1.3.b).
2. L'extraction des termes à partir de l'ontologie définie précédemment.
3. Identification des termes (voir le détail par la suite).

IV.2.2. Outils.

1. Un ordinateur ayant les caractéristiques suivantes :
   - Un processeur Intel Core I3.
   - Une fréquence de 2 GHz.
   - Un disque dure de 500 GO.
   - Une RAM de 3 GB.
2. Un système d'exploitation : Windows 7vin Edition Professional.
3. Un outil de développement Delphi 7.
4. Wordnet pour définir la sémantique des termes ou concepts.
5. Protégé pour exploiter l'ontologie.

IV.2.3. Présentation de l'application « RetroWebOnto ».

Voici l'interface d'accueil de notre application qui permettra de déclencher d'autres interfaces, chacune contienne un noyau de traitement spécifique à chaque étape :

Figure IV.2.3 : Interface d'accueil de l'application.

IV.2.3.1. Extraction de données à partir de la page HTML.

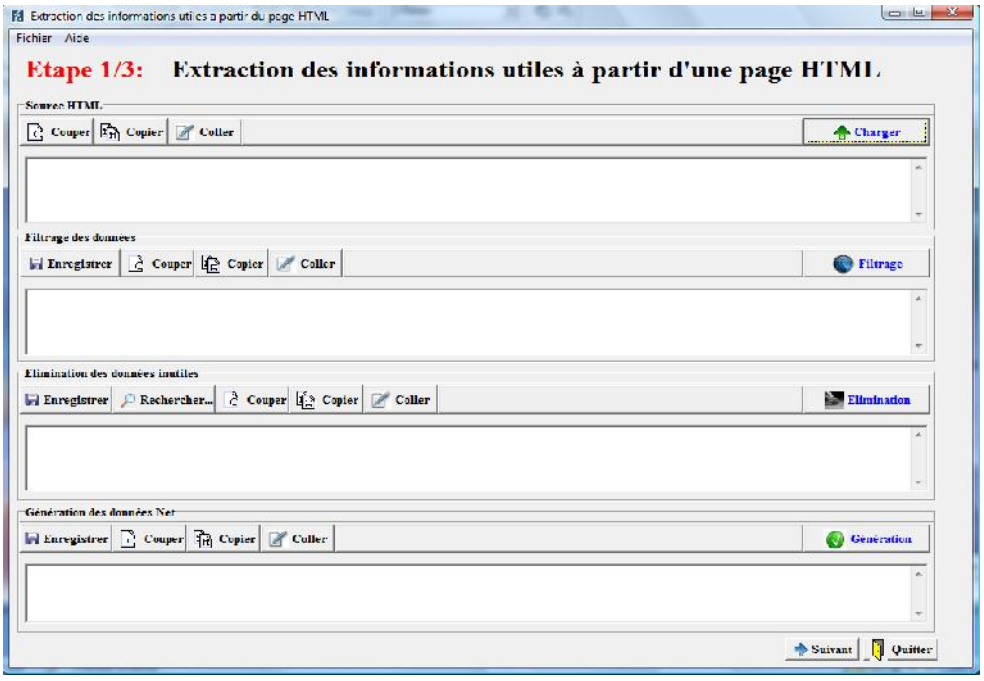


Figure IV.2.3.1 : Interface principal d'extraction des informations utiles à partir d'une page HTML

La première opération faite par cette interface est de permettre à l'usager de charger la source HTML d'une page et l'afficher dans un champ mémo qui acceptera les traitements d'édition (couper, copier et coller). En outre cette interface permet l'exécution d'autres opérations telles que : filtrage, élimination et génération.

IV.2.3.1.1. Filtrage de données.

Dans cette phase, nous avons utilisé un algorithme permettant d'éliminer toutes les balises de la source HTML, c'est-à-dire les balises ouvrantes et les balises fermantes.

```
Si le champ contient la source HTML vide
        Alors   Afficher message « charger la source HTML »
        Sinon
            Booléen est vrai
            Pour Chaque lignes Faire
                Pour chaque caractère (C) de la ligne Faire
                  Si C=< Alors Booléen est faux
                    Si l'objet Oi n'est pas vide Alors Ajouter l'objet Oi dans le champ de (Filtrage de
                données)
                      Fin si
                Fin si
                Si C=> Alors Booléen est vrai
                Fin si
                Si Booléen est Vrai alors Assembler les caractères pour obtenir un objet Oi
                Fin si
                Fin pour
                Ajouter l'objet Oi dans le champ de (Filtrage de donnée)
            Fin pour
Fin si
```

Dans la suite, vous trouverez un exemple d'exécution de l'opération de filtrage (voir page suivante) :

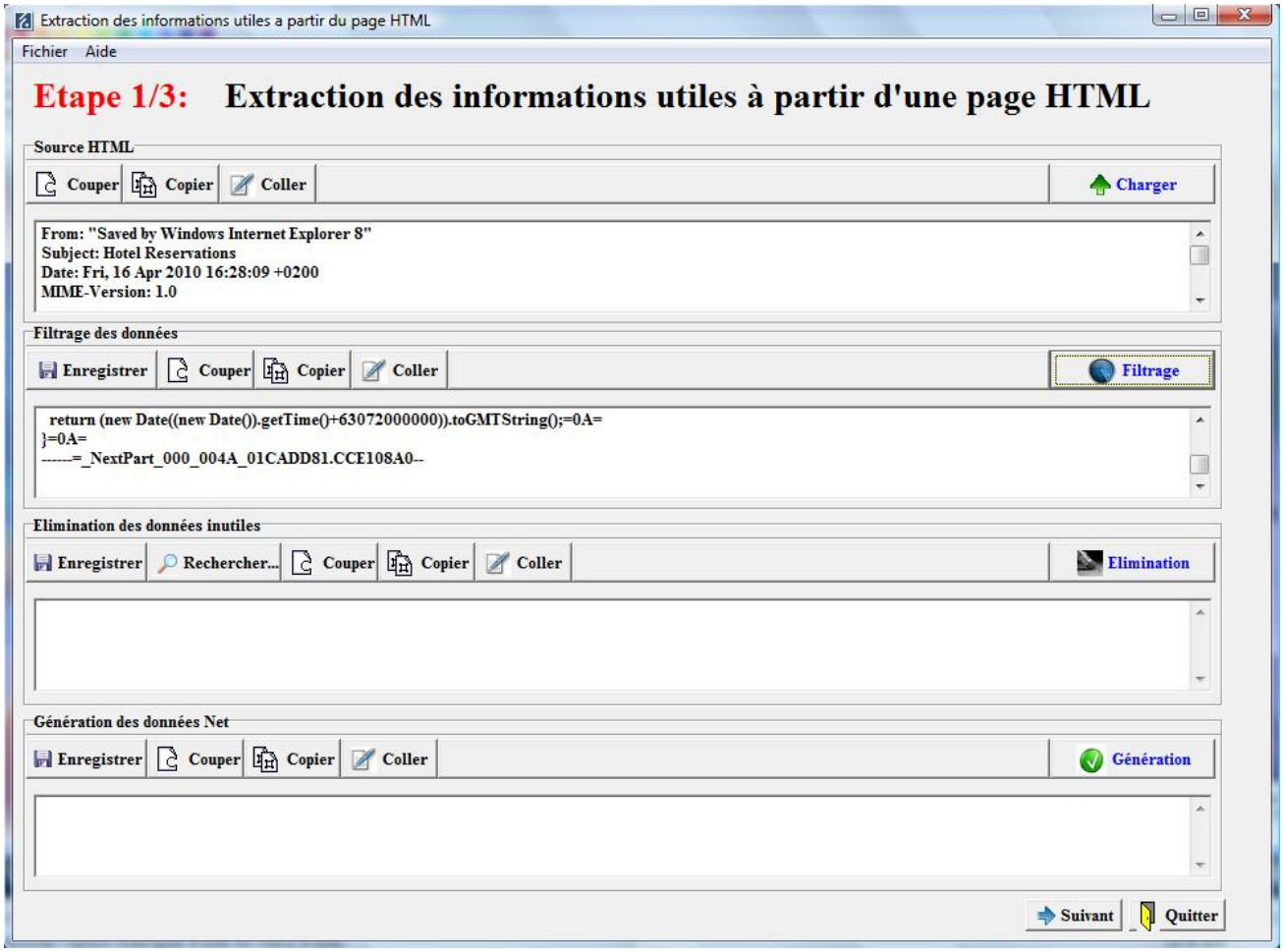


Figure IV.2.3.1.1 : Exemple de filtrage de données.

IV.2.3.1.2. Elimination de données inutiles.

Dans la deuxième phase, nous éliminons, surtout, les symboles spéciaux : +, &, _,), …etc.qui construisent l'interface de la page HTML. L'algorithme suivant exécute cette phase.

```
Pour Chaque ligne de la source HTML qui a été déjà filtrée Faire
    Si La ligne ne contient pas une des symboles suivants (hors balise) (+, =, (, ),……ect) Alors
        Ajouter l'objet Oj dans le champ de (Elimination de données inutiles)
    Finsi
Finpour
```

Dans la suite, un exemple sur l'élimination de données inutiles (voir la figure dans la page qui suit).

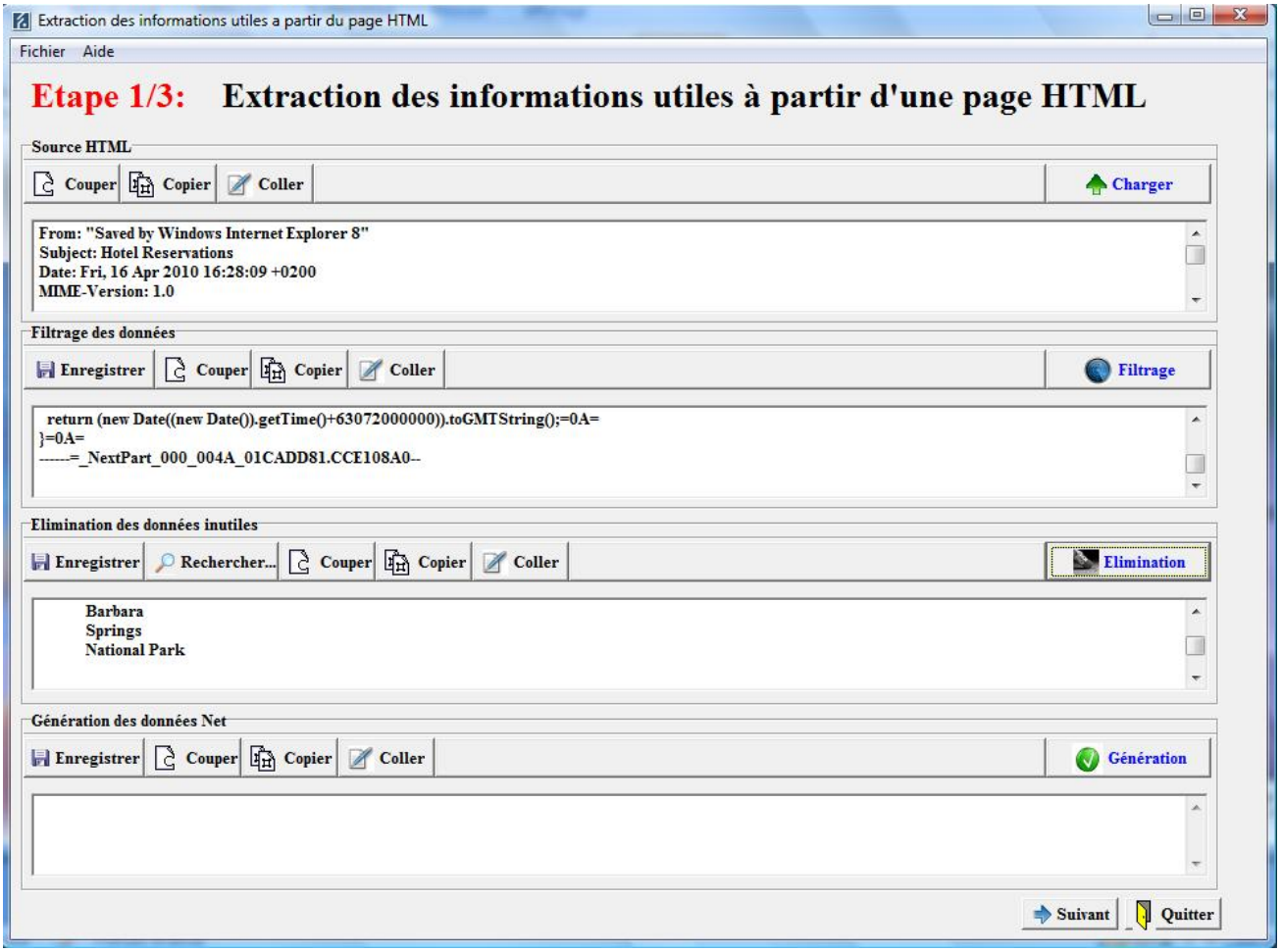


Figure IV.2.3.1.2 : Elimination de données inutiles.

IV.2.3.1.3. Génération de données Net.

Cette phase est la suite de la phase précédente, c'est-à-dire dans la partie Elimination nous avons fait la recherche des termes pour vérifier s'il est un Concept ou non. Si ce terme recherché n'est pas un concept on doit le supprimer par contre, on passe à la génération de données net.

Voici un exemple après la phase génération (voir la figure dans la page suivante).

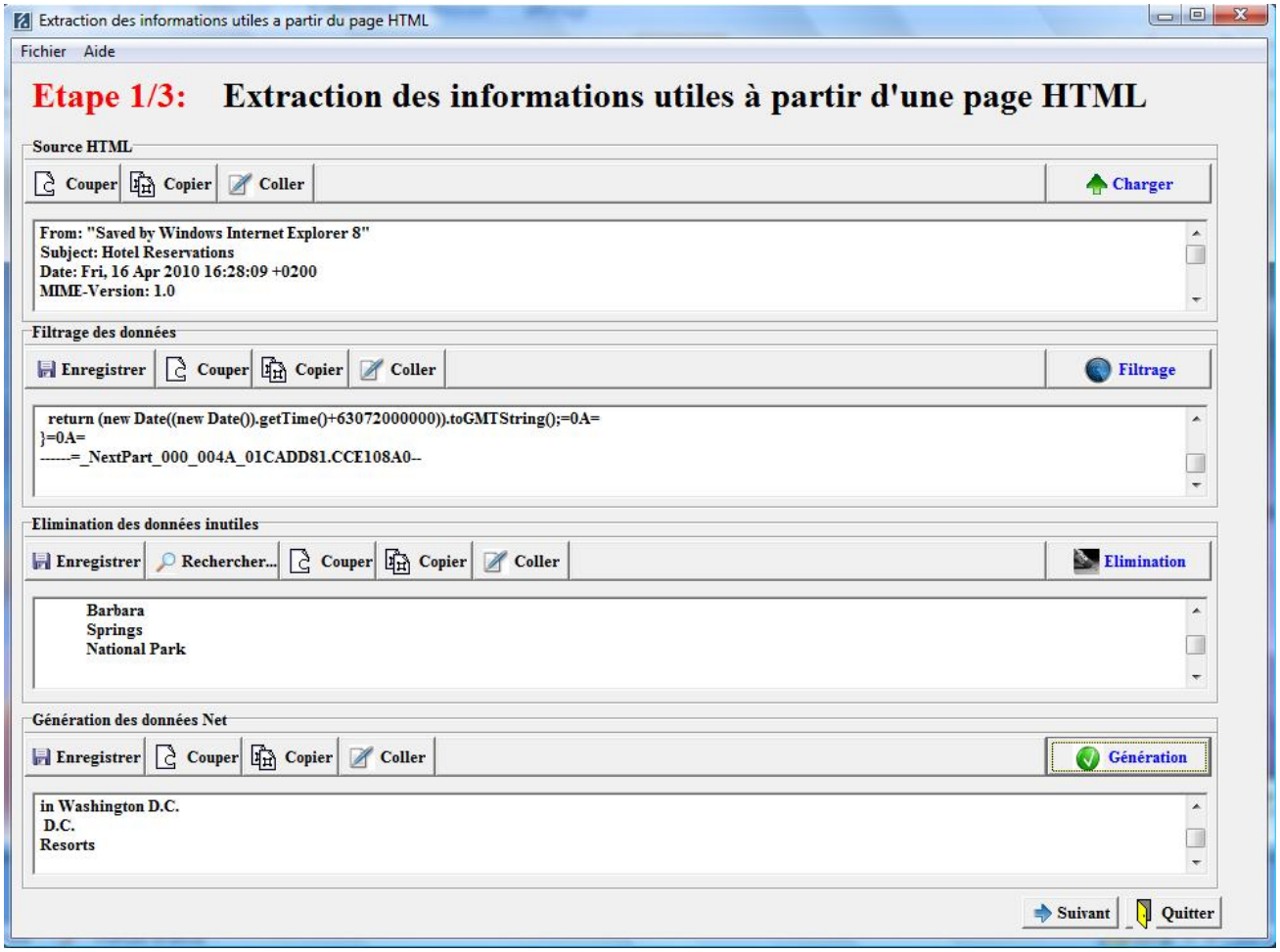


Figure IV.2.3.1.3 : Génération de données net.

IV.2.3.2. Extraction des concepts à partir de l'ontologie.

Cette étape permettra l'utilisation de l'ontologie en la chargeant dans un champ mémo, ce qui nous aidera à faire des traitements d'édition et par la suite faire une extraction des concepts pour pouvoir par la suite identifier les termes extraits des pages HTML.

L'interface de cette étape est illustrée dans la page suivante :

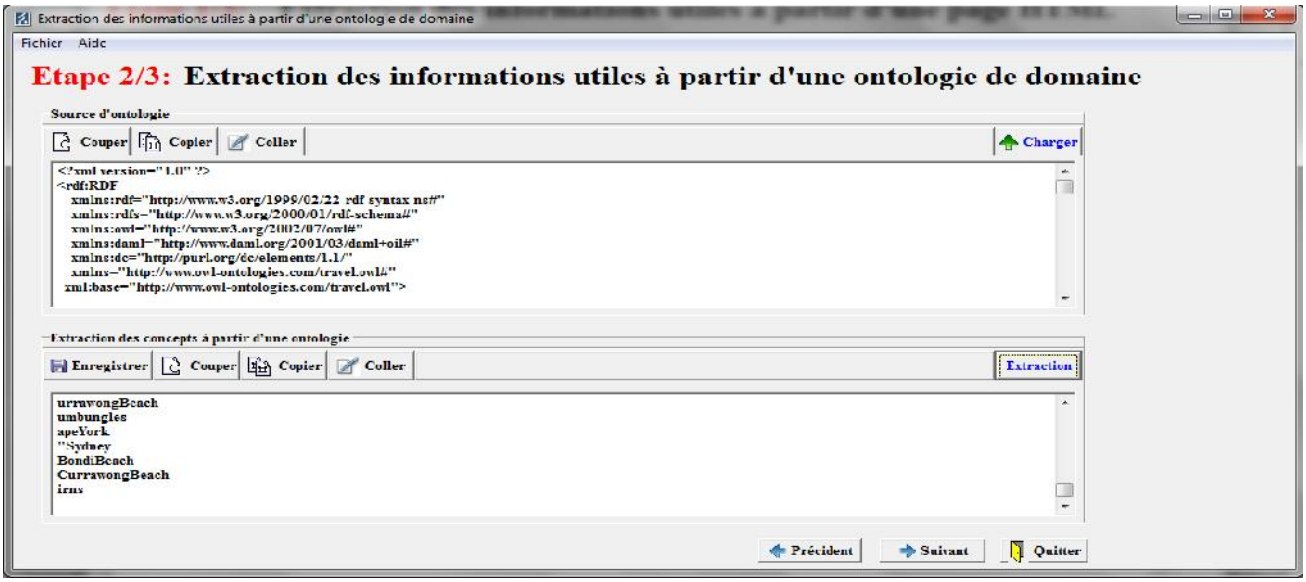


Figure IV.2.3.2. : Interface d'extraction des informations utiles à partir d'une ontologie.

IV.2.3.3. Identification.

Notre dernière étape d'implémentation est l'identification. Cette étape permettra d'identifier tous les termes issus de l'extraction à partir des sources HTML en utilisant les concepts extraits de l'ontologie par une intermédiaire qui est wordnet, cette dernière nous aidera à donner la sémantique des termes par un ensemble de synonymes (synset).

L'interface décrivant cette étape est la suivante :

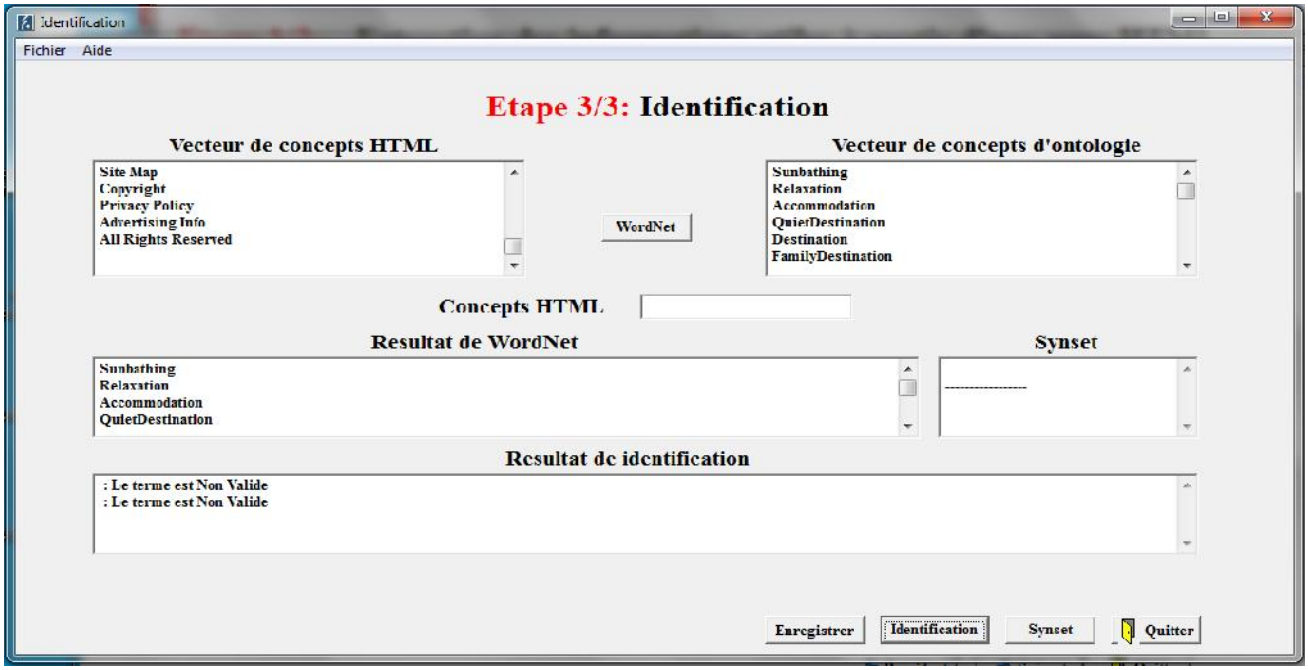


Figure IV.2.3.3. : Interface d'identification.

On remarque dans cette interface l'affichage de :

- Une matrice qui contienne deux lignes, la 1ère va contenir tous les concepts de l'extraction HTML, la 2ème va être remplie à la fin de l'identification par deux résultats : valide ou non valide.
- Un vecteur qui va contenir les concepts de l'ontologie.
- Un appel de wordnet par le biais d'un bouton qui permet de rechercher l'ensemble des synonymes (synset).
- Résultats issus de wordnet avec les synset.
- Résultat final de l'identification qui valide ou non le terme.

L'algorithme permettant l'exécution de cette étape est le suivant :

```
Charger les termes issus de l'extraction de la page HTML dans une matrice M.
A partir de WordNet, charger les têtes des Synsets avec leur valeur de similarité dans une matrice W.
A partir de Wordnet, charger les synonymes de chaque tête de synset dans un vecteur de vecteur V1.
Charger les termes issus de l'extraction de l'ontologie dans un vecteur V2.
    Pour chaque terme Ti du vecteur V1 Faire
        Pour chaque terme Oj du vecteur V2 Faire
            Comparer les termes Oj avec les termes Ti un par un.
            Si pour un terme Oj il ya un synonyme dans le vecteur V1 alors
                La tête de Synset dans la matrice W est Valide
            Si non la tête est non valide ;
            Fin si
        Fin pour
    Fin pour
```

IV.3. Conclusion.

L'approche de rétro-ingénierie se compose de quatre grandes étapes : extraction, identification, enrichissement et conceptualisation. Pour notre travail, nous avons atteint les deux premières étapes en essayant de les détailler en phases telle que mentionné dans ce chapitre.

Conclusion Générale

Le but de ce mémoire est de proposer une approche de rétro-ingénierie des applications Web en utilisant une ontologie de domaine pour générer un schéma conceptuel UML décrivant cette application. On a pris comme étude de cas le domaine touristique qui est un domaine pratiquement abandonné.

Notre contribution se situe dans le contexte de reprise d'existant et se focalise sur des applications Web incluant des pages dynamiques avec un contenu qui se change continuellement.

Dans notre approche nous considérons les ontologies comme la source sémantique principale qui permet d'identifier les concepts cachés dans les pages HTML. Deux autres phases de cette approche de rétro-ingénierie ne sont pas abordées dans ce mémoire qui sont la déduction ou l'inférence d'autres concepts en utilisant toujours l'ontologie puis la génération d'un nouveau schéma conceptuel (UML par exemple). Ces deux dernières phases pourront faire l'objet d'un autre sujet de mémoire.

Le système actuel ne représente que l'aspect statique de l'application Web. Il doit être étendu pour qu'il puisse décrire aussi l'aspect dynamique.

Pour le moment, le système ne peut analyser que des sites anglais. Ce problème peut être résolu par l'utilisation d'une base de données lexicale multilingue autre que WordNet.

Dans la version actuelle, on se restreint sur les tableaux et les listes comme informations utiles, on peut enrichir notre système par d'autres sources d'information tel que les formulaires, les liens hypertextes…etc.

A la fin, nous espérons que ce travail a été bien abordé et qui sera un point de départ pour d'autres travaux dans ce même contexte de rétro-ingénierie

Liste des tableaux et figures.

Liste des acronymes.

Acronyme	Désignation …	N° Page (s)
IA	Intelligence Artificielle	2
IC	Ingénierie des Connaissances	7
SBC	Systèmes à Base de Connaissances	7
W3C	World Wide Web Consortium	8,21,22,24 à 26,30,41
CNRS	Centre National de la Recherche Scientifique	15
OWL	Ontology Web Language	21 à 30
SKOS	Simple Knowledge Organisation System	21,26
XML	eXtensible Markup Language	21 à 24,36,45,46
RDF	Resource Description Framework	21 à 24,26,28
HTML	HyperText Markup Language	22,35 à 37,43,45 à 51,54,60 à 63,73,76 à 79,81 à 84
URI	Uniform Resource Identifier	22,24
RDFS	Resource Description Framework Schema	22,29
OWL DL	Ontology Web Language Description Logics	23,24,54,63
ISO	L'Organisation internationale de normalisation	24
LGI2P	Laboratoire de Génie Informatique et d'Ingénierie de Production	24
SWAD	Semantic Web Advance Development	26
OWLViz	Ontology Web Language Vision	28
RACER	Renamed ABox and Concept Expression Reasonner	28
WebApp	Web Application	31
NCSA	National Center for Supercomputing Applications	32
CERN	Centre Européen pour la Recherche Nucléaire	32
CGI	Common Gateway Interface	32
TCP	Transmission Control Protocol	33,37,38,42
IP	Internet Protocol	33,37,38,42
SGBD	Système de Gestion de Base de Données	34
HTTP	HyperText Transfer Protocol	35 à 38,45,73
PHP	Hypertext Preprocessor	35,41
ASP	Active Server Pages	35
SQL	Structured Query Language	35,41
OSI	Open System Interconnexion	37
HTTPS	HyperText Transfer Protocol Secured	38
XHTML	eXtensible HyperText Markup Language	39,45
CSS	Cascading Style Sheet	41
DMZ	Zone démilitarisée	42
ISP	Internet Service Provider	42
ADSL	Asymmetric Digital Subscriber Line	42
PSTN	Public Switched Telephony Network	42
RNIS	Réseau Numérique à Intégration de Services	42
PC	Personal Computer	42
WML	Wireless Markup Language	43
J2EE	Java Enterprise Edition	45
API	Application Programming Interface	45,50

JDBC	Java DataBase Connectivity	45
JSP	Java Server Pages	35,45
DTD	Document Type Definition	46
UML	Unified Modeling Language	47,49,51,55,59,60,62, 84
RMDM	Relationship Management Data Model	47
RMM	Relationship Management Methodology	47
DOM	Document Object Model	50,51
PPC	Proche Père Commun	52
RAM	Random Access Memory	76

C.1. Introduction et définitions.

WordNet et un thesaurus[1] pour la langue anglaise basé sur des études psycholinguistiques et développé à l'université de princeton par G.miller. Il a été conçu comme une ressource informatique qui couvre des catégories lexico-sémantique appelées synsets. Les synsets sont des ensembles de synonymes qui regroupent des items lexicaux ayant des significations similaires comme par exemple les mots "a board" (un panneau) et "a plank" (une planche) groupés dans le synset {board,plank}. Mais "a board" peut aussi désigner un groupe de personnes (un conseil d'administration par exemple) et pour désambiguïser ces significations homonymiques "a board" appartiendra aussi au synset {board,commitee}. La définition des synsets varie du très spécifique au très général. Les synsets les plus spécifiques ne regroupent qu'un nombre restreint de significations.

L'organisation de WordNet à travers des significations lexicales au lieu d'utiliser des unités lexicales le rend différent des dictionnaires traditionnels et des thesaurus. L'autre différence que présente WordNet par rapport aux dictionnaires traditionnels se traduit par la séparation des données en quatre bases de données associées aux catégories de verbes, de noms, d'adjectifs et d'adverbes. Ce choix d'organisation est motivé par des recherches psycholinguistiques sur l'association de mots aux catégories syntaxiques par des sujets humains. Chaque base de données est organisée différemment des autres. Les noms sont organisés en hiérarchie, les verbes par des relations, les adjectifs et les adverbes par des hyper-espaces N-dimension [Miller and al-1990].

C.2. Les bases de données WordNet.

C.2.1. La base des noms.

Dans cette base, on distingue un ensemble d'entrées uniques et les"top types "certains synsets ne sont couverts par aucun autre synset ; chacun d'eux consiste une hiérarchie séparée correspondant à une distinction relative des champs sémantiques. On trouve 25 types de synsets organisés en une hiérarchie comme (voir page suivante) :

[1] Un thesaurus est un vocabulaire de termes contrôlés d'indexation, structuré de manière à ce qu'il mette en évidence les relations *a priori* entre les concepts. Comme une liste de mots-clés, c'est un instrument qui utilise une terminologie normalisée et contribue à aider l'utilisateur à sélectionner de manière organisée des occurrences dans une base de données [Olfa.-2003].

{plant, flora}

{living thing, organism} {animal, fauna}

{person, humain being}

{thing, entity}

{naturel object}

{artifact}

{non-living thing, objet} {substance}

{food}

C.2.2. La base des verbes.

WordNet comporte environ 21000 formes de verbes dont 13000 sont des entrées uniques, 8400 de significations de verbes (synsets) incluant des expressions phrastiques comme "look up" et "fall back" [Olfa.-2003]. Au départ, les verbes ont été regroupés sur des critères généraux en 17 grandes familles comme le mouvement, la possession, la perception, le contact, la communication, le changement, l'apprentissage, la conception, la création, l'émotion, le soin du corps et la vitalité, les relations et les interactions sociales et la météorologie. Puis ces classes ont été subdivisées jusqu'à obtenir un synset. Entre les différents synsets, il existe des relations lexicales de nature pragmatique :

- L'implication, ronfler implique dormir.
- La troponomie, une relation générique qui fait que les deux verbes crier et parler sont liés par cette relation car crier peut être vu comme parler très fort.

La figure suivante (voir page suivante) montre la hiérarchie des concepts que l'on trouve dans WordNet pour le terme humain.

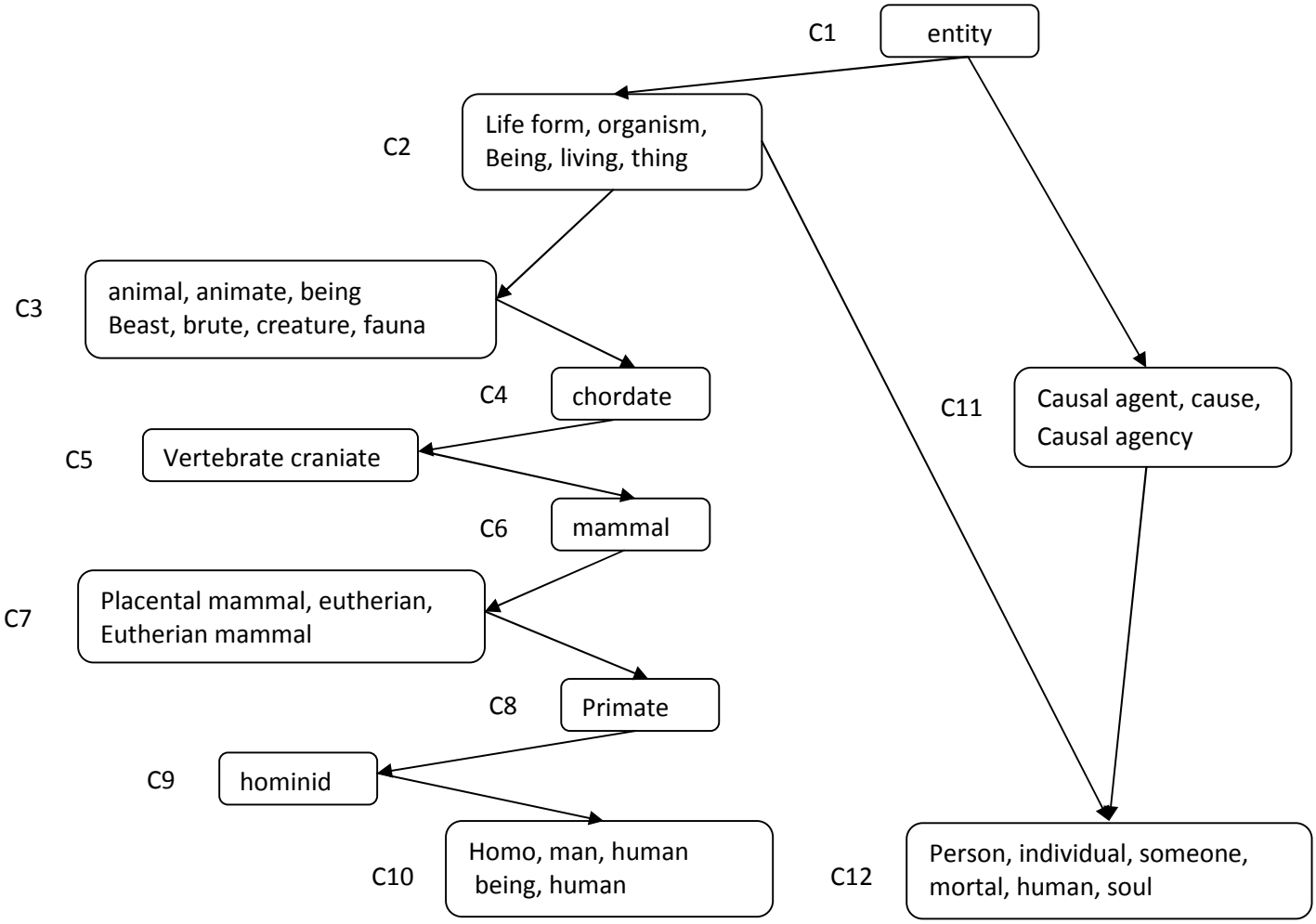


Figure C.2.2 : Hiérarchie WordNet pour le terme human.

Human a deux sens dans WordNet représentés par les concepts C10 (homo, human being, human) et C12 (person, individual, someone, mortal, human soul). Le concept le plus haut dans la hiérarchie représentée est le terme entity. On peut constater que les deux sens de human ont un ancêtre commun qui est le concept C2 (life form, organism, being, living, thing). Le sens C12 hérite à la fois de C2 et de C11 (causal agent, cause, causal agency).

C.3. Relations sémantiques dans WordNet.

La liste qui suit énumère les relations sémantiques disponibles dans WordNet. Ces relations concernent des concepts, mais les exemples que nous donnons sont basés sur des mots.

- **Synonymie :** relation liant deux concepts équivalents ou voisins (frêle / fragile). Il s'agit d'une relation symétrique.
- **Antonymie :** relation liant deux concepts opposés (petit / grand). Cette relation est symétrique.

- **Hyperonymie :** relation liant un concept- 1 à un concept-2 plus général (tulipe /fleur).
- **Hyponymie :** relation liant un concept- 1 à un concept-2 plus spécifique. C'est la réciproque de l' hyponymie. Cette relation peut-être utile en recherche documentaire. En effet, si l'on cherche tous les textes traitant de véhicules, il peut être intéressant de retrouver ceux qui parlent de voiture ou de motos. Des expériences utilisant la relation d' hyponymie sont présentées dans la section 9.3.7.
- **Méronymie :** relation liant un concept-1 à un concept-2 qui une de ses parties (fleur / pétale), un de ses membres (foret / arbre) ou une substance le constituant (vitre / verre).
- Métonymie : relation liant un concept-1 à un concept-2 dont il est une des parties. C'est la relation inverse de la méronymie.
- **Implication :** relation liant un concept-1 à un concept-2 qui en découle (marcher /faire un pas).
- **Causalité :** relation liant un concept-1 à son effet (tuer / mourir).
- **Valeur :** relation liant un concept-1 (adjectifs) qui est un état possible pour un concept-2 (pauvre / condition financière).
- **A pour valeur :** relation liant un concept-1 à ses valeurs (adjectifs) possibles (taille /grand). C'est la relation inverse de valeur.
- **Voir aussi :** relation entre des concepts ayant une certaine affinité (froid / gelé).
- **Similaire à :** certains concepts adjectifs dont le sens est proche sont regroupés. Un synset est alors désigné comme étant central au regroupement. La relation similaire à lie un synset périphérique au synset central (moite /humide).
- **Dérivé de :** indique une dérivation morphologique entre le concept cible (adjectif) et le concept origine (froidement / froid).

C.4. Conclusion.

WordNet est un système de référence lexical en ligne dont la conception est fortement inspirée des théories psycholinguistiques récentes de la mémoire sémantique humaine. Les noms, les verbes, les adjectifs et les adverbes anglais sont organisés en ensemble de synonymes (synset), chacun représente un concept lexical.

Différentes relations lient les synsets sous forme de réseau sémantique [George and al.-1993].

WordNet peut être utilisé autant qu'un dictionnaire ordinaire. Aussi, elle peut être utilisée dans des applications plus complexes telles que le calcule de distance sémantique.

Liste des algorithmes

Références bibliographiques

[Alexander-2003] Alexander Maedche, ONTOLOGY LEARNING FOR THE SEMANTIC WEB. Pages 122, 123. 2003.

[Antoniol and al.-2000] Antoniol G., Canfora G., Casazza G., and De Lucia A., Web Site Reengineering Using RMM. 2nd nd International Workshop on Web Site Evolution. March 1, 2000.

[BACHIMONT 2000] « Engagement sémantique et engagement ontologique : conception et réalisation d'otologies en ingénierie des connaissances ». In : CHARLET, Jean et al. Ingénierie des connaissances. Évolutions récentes et nouveaux défis. Paris : Eyrolles, p. 305-323.

[Banerjee and al.-2003] Banerjee, S., and Pedersen, T., Extended gloss overlaps as a measure of semantic relatedness. In Proceedings of the Eighteenth International Joint Conference on Artifi-cial Intelligence, Pages 805–810. 2003.

[BANEYX 2007] Construire une ontologie de la pneumologie : Aspects théoriques, modèles et expérimentations. Thèse de doctorat en informatique médicale. Paris 6.

[BERNERS-LEE 2001] The Semantic Web, Scientific American, http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

[CHANDRASEKARAN et al.1999] What are ontologies and why do we need them? IEEE Intelligent Systems. 14(1):20-26.

[CHARLET et al. 2004] Ontologies pour le Web sémantique. In Revue i3, numéro Hors Série « Web sémantique ».

[Chung and Lee.-2000] Sam Chung and Yun-Sik Lee. Reverse Software Engineering with UML for Web Site Maintenance. 2000.

[CLAUBERG 1647] Elemanta philosophiae ; sive Ontosophia…Groningae : typis J. Nicolai, 311 p.

[Conallen-1999a] J. Conallen. Building Web Application with UML. Object technology. Addison-Wesley Longman, Reading, Massachusetts, USA, first edition, Dec. 1999.

[Conallen-1999b] J. Conallen. Modeling Web Application Architectures with UML. Communications of the ACM (v 42, n 10). October 1999.

[Di Lucca and al.-2001] Di Lucca G.A., Di Penta M., Antoniol G., Casazza G., An Approach for Reverse Engineering of Web-Based Applications.2001.

[Fabrice and al.-2003] Fabrice Estiévenart., Aurore François., Jean Henrard., Jean-Luc Hainaut., A tool-supported method to extract data and schema from web sites.

[Filippot and Paolo.-2001] Filippo Ricca and Paolo Tonella. Analysis and Testing of Web Applications. 2001.

[FÜRST 2004] Contribution à l'ingénierie des ontologies : une méthode et un outil d'opérationnalisation. Thèse de doctorat en informatique. École polytechnique de l'Université de Nantes.

[George and al.-1993] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Introduction to WordNet: An On-line Lexical Database (Revised August 1993).

[GOCLENIUS 1613] Lexicon philosophicum. Francofurti : typis viduae M. Beckeri, 1143 p.

[GUARINO 1997a] Some organizing principles for a unified top-level ontology. Proceedings of the AIII Spring Symposium on Ontological Engineering.

[GUARINO 1997b] Understanding, building and using ontologies: a commentary to 'using explicit ontologies in kbs development.

[GUARINO et POLI 1995] Formal ontology in conceptual analysis and knowledge representation. Special issue of the International Journal of Human and Computer Studies. 43(5/6):625-640.

[GRUBER 1993a] Formal ontology in conceptual analysis and knowledge representation. Chap. Towards principles for the design of ontologies used for knowledge sharing. Kluwer Academic Publishers.

[Hassan and Richard.-2002] Ahmed E. Hassan and Richard C. Holt. Architecture Recovery of Web Applications. 2002.

[HEGEL 1994] Science de la logique. Paris : Aubier.

[Hirst and al.-1998] Hirst G., St-Onge D., Lexical chains as representations of context for the detection and correction of malapropisms. In Fellbaum, C., ed., WordNet: An electronic lexical database. MIT Press. Pages: 305–332.

[Jiang and a1.-1997] Jiang, J., and Conrath, D., Semantic similarity based on corpus statistics and lexical taxonomy. In Proceedings on International Conference on Research in Computational Linguistics, Pages: 19-33. 1997.

[KANT 2001] Critique de la raison pure. Paris : Flammarion, 749 p.

[LE ROBERT 2006] Le nouveau Petit Robert : dictionnaire alphabétique et analogique de la langue française. Dictionnaires Le Robert : Paris, 2837 p.

[Lin and al.-1998] Lin, D., An information-theoretic definition of similarity. In Proceedings of the International Conference on Machine Learning. 1998.

[Leacock and al.-1998] Leacock C., Chodorow M., Combining local context and WordNet similarity for word sense identification. In Fellbaum, C., ed., WordNet: An electronic lexical database. MIT Press. Pages: 265-283. 1998.

[Marc and al.-2004] Marc Ehrig, Peter Haase, Mark Hefke, Nenad Stojanovic, Similarity for Ontologies -a Comprehensive Framework. 2004.

[Miller and al-1990] MILLER George A.BECKWITH R.FELLBAUM C.GROSS D.MILLER K.J."Introduction to WordNet: an on-line lexical database", Journal of lexicography 3, PP.235-244.

[Olfa.-2003] Olfa jenhani. Ontologies pour le WEB: relations, construction d'ontologies et méthodes de raisonnement pour la génération de langue naturelle. INRIA- ARC GeNI- Mai 2003.

[Patwardhan and al.-2003] Patwardhan S., Banerjee S., Pedersen T., Using measures of semantic relatedness for word sense disambiguation. In Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics. Pages: 241–257. 2003.

[PSYCHE et al. 2003] « Apport de l'ingénierie ontologique aux environnement de formation à distance ». Revue Sticef, vol. 10. http://sticef.univ-lemans.fr/num/vol2003/psyche-06s/sticef_2003_psyche_06s.pdf

[Ranwez, 2000] Sylvie Ranwez. Composition de documents Hypermédia Adaptatifs à partir d'Ontologies et de requêtes Intentionnelle de l'Utilisateur, PhD thesis in computer science, Montpellier II University.

[RASTIER 1987] Sémantique interprétative. Paris : PUF,

[RASTIER 1995] « Le terme : entre ontologie et linguistique ». La banque des mots, n°7, p. 35-65

[RASTIER et al. 1994] Sémantique pour l'analyse. Paris: Masson,

[Resnik and al.-1995] Resnik P., Using information content to evaluate semantic similarity in a taxonomy. In Proceedings of the 14th International Joint Conference on Artificial Intelligence. Pages: 448–453. 1995.

[SOWA 1999] Knowledge representation: Logical, philosophical and computational foundations. Brooks Cole Publishing Co.: Pacific Grove, CA USA.

[USCHOLD et GRUNINGER 1996] Ontologies: Principles, Methods and Applications". Knowledge Engineering Review, vol.11, n°2, p. 93-136

[USCHOLD et KING 1995] Towards a methodology for building ontologies. In Workshop on Basic Ontological Issues in Knowledge Sharing. International Joint Conference on Artificial Intelligence.

[VAN HEIJST et al.] Using explicit ontologies in KBS development. International Journal of Human-Computer Studies, 45(2/3), 183-292.

[W3C 2004] Michael K. Smith, Chris Welty, Deborah L. McGuinness. OWL Web Ontology Language - Reference. http://www.w3.org/TR/2004/REC-owl-ref-20040210/ (en ligne au 16 juin 2005).

[WEL 01] WELTY C. & GUARINO N., Supporting ontological analysis of taxonomic relationships, *Data et Knowledge Engineering* (39), pages 51-74, 2001.

[WOLFF 1730] Philosophia Prima sive Ontologia. Francofurti et Lipsiae : prostate in officina libraria Rengeriana, 696 p.

[Wu and al.-1994] Wu Z., Palmer M., Verb semantics and lexical selection. In 32nd Annual Meeting of the Association for Computational Linguistics, Pages: 133–138. 1994.