Université Mouloud Mammeri de Tizi-Ouzou.

Faculté de Génie Electrique et D'informatique.

Département D'informatique.

MEMOIRE DE FIN D'ETUDE

En vue d'obtention du diplôme de Master 2 en Informatique.

Option : Conduite de projets informatique

Thème :

Ontologie de domaine pour un web sémantique destiné au e-Learning

Réalisé par :

HAMDI Ahmed

Membres du jury:

1. Président : M^me^ SINI Ghenima.
2. Examinateur 1 : M^me^ LAGAB Aoudjit.
3. Examinateur 2 : M^r^ CHAIEB Yazed.
4. Promoteur :M^me^ BOUARAB-DAHMANI Farida.

2011

Remerciements

Je tiens à exprimer ma profonde gratitude à ma promotrice, Madame BOUARAB Farida pour m'avoir proposé ce sujet, m'avoir encadrées durant cette année, ainsi que pour ses conseils judicieux.

Mes plus vifs remerciements pour les membres du jury d'avoir accepté d'honorer par leur jugement notre travail.

Enfin je remercie tous ceux qui ont contribué pour la réalisation de ce travail.

Sommaire

Chapitre I : Généralité sur les ontologies, web sémantique et E-Learning.

Chapitre II : Construction de l'ontologie

Chapitre III :Exploitation de l'ontologie

Liste des figures

Liste des figures :

Introduction générale

Introduction générale :

Dans les années 90, les travaux menés dans le domaine de la Représentation de Connaissances ont mis en évidence la faisabilité de modéliser explicitement et de façon consensuelle les caractéristiques structurelles et descriptives du domaine sur lequel porte une connaissance pour rendre celle-ci plus facilement partageable. Ces travaux ont abouti à la définition de modèles permettant d'expliciter la sémantique des données appelées ontologies **[GRU, 93].** Des travaux utilisant des ontologies ont été menés par déférentes communautés pour proposer des solutions à des problèmes très variés. Ces travaux ont vu naître :

Différentes catégories et formalismes d'ontologies.

Comment représenter les dites ontologies, quels outils et quelles méthodes pour les exploiter.

L'un des domaines d'application les plus courant des ontologies est celui du Web, en effet, depuis les années 2000, dans le contexte du Web Sémantique, les ontologies ont été utilisées pour annoter les données du Web afin de faciliter leur traitement par les machines et de permettre la communication entre ces machines et des agents humains selon une base sémantique.

L'apprentissage en ligne, ou bien le E-Learning, est parmi les domaines, ou les ontologies et le web sémantique jouent un rôle très important car Le Web sémantique permet le développement des ontologies et l'annotation des ressources d'apprentissage, son utilisation devient alors très utile et adéquate pour implémenter un système E-Learning.

Pour mettre en évidence ces différents concepts, et dans le cadre de notre travail nous avons opté pour le plan suivant :

Apres une introduction générale, nous consacrerons le premier chapitre au généralité sur les ontologies, le web sémantique et le E-Learning ou nous allons définir chacun d'eux, donner leurs propriétés et montrer l'interaction qui existe entre eux pour entamer le deuxième chapitre ou nous allons définir notre ontologie de domaine d'enseignement des base de données relationnelle, l'édité sous l'éditeur d'ontologie Protégé et vers la fin générer le code OWL correspondant à la base de connaissance obtenu pas l'ontologie.

Une fois le code OWL est obtenu, dans le troisième chapitre nous allons passer à son exploitation sous l'éditeur Protégé et sous un environnement Java.

Pour conclure après cette étude, nous présenterons les différentes perspectives de recherches.

CHAPITRE I

Généralité sur les ontologies, web sémantique et le l'e-Learning.

Introduction :

La nouvelle génération du Web, apparaît comme une technologie très prometteuse pour implémenter des systèmes e-Learning. Le Web Sémantique constitue un environnement dans lequel les agents humains et machines vont communiquer selon une base sémantique. Une des caractéristiques principales est la compréhension partagée basée sur un squelette d'ontologie. L'ontologie permet l'organisation du matériel d'apprentissage autour de petites pièces d'objets d'apprentissage sémantiquement annotés.

Une ontologie définit un vocabulaire commun pour les chercheurs qui ont besoin de partager l'information dans un domaine. Elle inclut des définitions lisibles en machine des concepts de base de ce domaine et de leurs relations, nous développant des ontologies pour plusieurs raisons dont nous citons quelques une :

- Partager la compréhension commune de la structure de l'information entre les personnes ou les fabricants de logiciels.
- Permettre la réutilisation du savoir sur un domaine.
- Expliciter ce qui est considéré comme implicite sur un domaine.
- Distinguer le savoir sur un domaine du savoir opérationnel.
- Analyser le savoir sur un domaine. »
- Capturer, modéliser, transformer une connaissance contextualisée.
- Présenter une organisation hiérarchique ou une taxonomie des concepts pertinents et des relations qui existent entre ces concepts.
- Traiter et automatiser les raisonnements de/sur cette connaissance.

Afin de présenter le contexte de notre travail, cette première partie tente de faire un récapitulatif sur les ontologies, le web sémantique et le e-Learning tout en abordant leurs éléments, leurs caractéristiques et leurs objectif ainsi les relations qui existe entre eux.

I. Les ontologies :

I.1 Origine et définition :

La philosophie et particulièrement la métaphysique générale, est à l'origine des ontologies. Aristote l'a définie comme étant la science de l'Être.

Pour GRUBER **[GRU 93]**, une ontologie est une spécification explicite et formelle d'une conceptualisation faisant l'objet d'un consensus. La conceptualisation fait référence à un modèle (au sens ensemble structuré) de concepts ce qui signifie que tous les types primitifs, les concepts et les contraintes qui sont utilisés dans les spécifications de l'ontologie sont explicitement définis. Une extension de cette définition considère qu'une ontologie est un système formel dont l'objectif est de représenter les connaissances d'un domaine spécifique au moyen d'éléments de base, les concepts, définis et organisés les uns par rapport aux autres. Elle est aussi considérée comme une collection de descriptions explicites, complète et consensuelles de concepts, en mettant l'accent sur la précision de leur sens.**[BOU, 07]**

I.2 Les éléments d'une ontologie :

La connaissance dans les ontologies est principalement formalisée en utilisant les cinq types de composants **[GRU 93]**: classes, relations entre concepts, fonctions, axiomes et instances. Sowa **[SOW 00]** rajoute à cette liste le composant rôle.

I.2.a Les classes ou Concepts:

Les connaissances portent sur des objets auxquels on fait référence à travers des concepts Qui sont habituellement organisés sous forme hiérarchisée dans l'ontologie.
Un concept peut représenter un objet matériel, une notion, une idée **[USK 95]** et peut être divisé en trois parties :
Un terme ou label qui est l'expression linguistique utilisée couramment pour y faire référence.
Une notion qui désigne, au sens de la représentation des connaissances, l'intension du concept. Elle contient sa sémantique qui est définie à l'aide de propriétés, d'attributs, de règles et de contraintes.

Un ensemble d'objets auxquels le concept fait référence forment l'extension du concept, autrement dit, ses instances.

Par exemple, le label de concept « hôpital » renvoie aussi bien à la notion d'hôpital en tant qu'endroit où les gens se soignent qu'à l'ensemble des objets de ce type : hôpital MUSTAPHA.

I.2.b Les relations:

Représentent un type d'interaction entre les concepts du domaine. Elles lient les concepts primitifs (ou simples) entre eux pour construire des représentations conceptuelles complexes. Elles sont formellement définies comme n'importe quel sous ensemble d'un produit de n-ensemble : R dans C1 x C2 x... x Cn.

Selon **[GUC, 95]** et **[KAS 02]**, les principales relations jugées utiles à la modélisation d'une ontologie sont: « instance-de », « sorte-de », « appartenance-a », «dépendance » et « subsomption (**Is-a) »**. Cette dernière est implicite et a un statut particulier car elle définit un lien de généralisation qui structure la hiérarchie ontologique.

On dit qu'un concept C1 (concept père) subsume un concept C2 (concept fils) si toute propriété sémantique de C1 est également une propriété sémantique de C2 et si C2 est plus spécifique que C1. L'extension du fils est donc forcément plus réduite que celle de son père mais son intension est plus riche. Par exemple, le concept «pathologie» subsume le concept «pneumonie».

I.2.c Les fonctions:

Sont un cas spécial de relations dans lesquelles le n ième élément de la relation est unique pour les n-1 éléments précédents. Formellement, des fonctions sont définies comme: F: C_1 x C_2 x... x $C_{n-1} \rightarrow C_n$, ou les Ci sont des concepts. Comme exemple de fonctions binaires il y a la fonction mère-de ou carré-de, comme fonction ternaire, le prix d'une voiture usagée sur lequel on peut se baser pour calculer le prix d'une voiture d'occasion en fonction de son modèle, de sa date de construction et de son kilométrage.

I.2.d Les axiomes (ou règles):

Les axiomes sont des expressions qui sont toujours vraies. Ils ont pour but de définir dans un langage logique la description des concepts et des relations permettant de représenter leur sémantique. Ils représentent les intentions des concepts et des relations du domaine et, de manière générale, les connaissances n"ayant pas un caractère strictement terminologique **[STA 00].** Leur inclusion dans une ontologie peut avoir plusieurs objectifs :

- Définir la signification des composants.
- Définir des restrictions sur la valeur des attributs.
- Définir les arguments d‟une relation.
- Vérifier la validité des informations spécifiées ou en déduire de nouvelles.

I.2.e Les instances:

Les instances constituent des valeurs concrètes et des occurrences pour les concepts et les relations par exemple les individus « Mourad » et «Samir» sont des instances du concept « personne ».

I.2.f Les rôles:

Une entité peut être caractérisée par un rôle **[SOW 00]** et ceci en définissant quelques rôles qu'elle peut jouer dans sa relation avec une autre entité. Par exemple, le type «Humain » est un type de phénomène qui dépend de la forme interne de l'entité ; mais la même entité peut être caractérisée par des rôles du type, Mère, Employé ou Piéton.

I.3 Caractéristiques fondamentales des ontologies [KAV 04] :

Les ontologies sont *formelles.* Ceci signifie qu'elles sont exprimées dans une langue qui a une syntaxe clairement définie et base mathématique pour leur signification. Comme les concepts sont exprimés formellement, ils peuvent être traités par des programmes informatiques. Les « concepts » ou les « objets » qui existent dans des techniques den modélisation traditionnelles (schéma relationnel et UML, par exemple) sont seulement semi formels. Elles ne peuvent donc pas être manipulées automatiquement par des logiciels sans un effort considérable (et coûteux) de programmation de manière à faire ressortir leurs significations.

Les ontologies sont *lisibles par les humains*. Ceci signifie qu'elles peuvent être développés, partagés, et compris non seulement par des programmes informatiques, mais aussi par les communautés d'experts de domaine ainsi que des utilisateurs potentiels.

Les ontologies sont *vastes*. Elles sont conçues avec le but d'inclure toute la Signification appropriée des concepts liés à un domaine ; pas simplement celles requise pour une application particulière. Cela veut dire que si toute la signification des concepts est capturée par une ontologie, elle peut être comprise, modifiée, et contrôlée par n'importe quel expert de domaine.

Les ontologies sont *partageables*. Ils sont construits sur la base de bibliothèques communes de concepts fondamentaux et sont utilisables à travers de multiples domaines d'application. Ceci facilite la combinaison des ontologies développées séparément pour permettre la communication entre les systèmes d'information qui doivent partager des informations basées sur des concepts communs.

I.4 Les types d'ontologies :

I.4.a Selon le degré de formalisme :

[USG, 96] ont identifié quatre types d'ontologies: les ontologies informelles, les ontologies semi-formelles et les ontologies rigoureusement formelles.

1. Les ontologies hautement informelles : exprimées en langage naturel.

2. Les ontologies semi-informelles : elles sont exprimées sous une forme limitée, restreinte et structurée du langage naturel (en utilisant des modèles), c'est à dire des patrons ont été mis en œuvre.

3. Les ontologies semi-formelles : exprimées dans un langage défini artificiellement et formellement.

4. Les ontologies rigoureusement formelles : exprimées dans un langage contenant une sémantique formelle, des théorèmes et des preuves de propriétés telles que la robustesse, l'exhaustivité, la complétude et la consistance.

Selon se type de classification nous pouvons classer notre ontologie que nous allons voir plus tard dans les ontologies semi-formelles. Car elle est exprimée dans un langage bien défini et bien formel qui est le OWL.

I.4.b Selon les objets modélisés :

Les ontologies ont été aussi regroupées dans **[PEL, 04]** en se basant sur les objets modélisés par l'ontologie afin de répondre à un but précis

1. Les ontologies Supérieures : dites aussi de haut niveau « top level ontologies ou Upper Level ontologies» **[GUA, 98],** ontologies « génériques ou noyaux d'ontologies » «méta-ontologie », «ontologies de sens commun/général », elles sont universelles, réutilisables et référençable à partir des concepts des autres niveaux d'ontologies . Elles comportent des concepts abstraits (généraux) subsumant les concepts existant dans les différents domaines. Une ontologie de haut niveau est généralement conçue afin de réduire

les incohérences des termes définis plus bas dans la hiérarchie. Il n'existe pas pour le moment d'ontologies de haut-niveau unifiées.

2. Les ontologies de domaine : ce type décrit un vocabulaire appartenant à un domaine générique donné tel que la médecine. Elles ne sont pas propres à une tâche précise et présentent une bonne précision et se rapportent à un certain type d'artefacts.

Et c'set dans ce domaine que nous pouvons classer notre ontologie qui décrit le domaine d'enseignement et plus précisément les bases de données relationnelles.

3. Les ontologies de tâche : spécifiques à une tâche générique, telle que la vente, et indépendamment du domaine d'application.

4. Les ontologies d'application : Correspondent à l'exécution d'une tâche particulière et leur domaine d'application est restreint. Elles sont souvent des spécialisations des ontologies de domaine et des ontologies de tâche **[GUA 98].**

5. Ontologies de représentation: Ce type d'ontologies est un cas particulier d'ontologies supérieures qui regroupe des concepts déjà utilisés pour formaliser les connaissances. Indépendamment des domaines **[GUC, 94]** puisqu'elles décrivent des primitives cognitives communes. Parmi les ontologies de représentation, on trouve la « Frame-Ontology » qui définit, de manière formelle, les concepts utilisés principalement dans des langages à base de frames : classes, sous-classes, attributs, valeurs, relations et axiomes **[GRU 93]**.

6. Les ontologies de raisonnement : regroupe les processus de raisonnement appliqués aux connaissances qui forment eux-mêmes un domaine de connaissance. On parle particulièrement d'ontologies développées pour représenter des connaissances génériques mises en œuvre lors de la résolution automatique de problèmes. Un exemple de cette catégorie d'ontologies a été concrétisé grâce aux travaux de **[CHA, 98].**

I.4.c Selon la granularité :

La classification suivante est en fonction du degré de granularité, c'est-à-dire quel niveau de détail des objets de la conceptualisation est préconisé. En fonction de l'objectif opérationnel, une connaissance plus ou moins fine du domaine est nécessaire et des propriétés considérées comme accessoires dans un contexte peuvent se révéler indispensables dans un autre :

1. Granularité fine : correspondant à des ontologies très détaillées, possédant ainsi un vocabulaire plus riche capable d'assurer une description détaillée des concepts pertinents d'un domaine ou d'une tâche **[FUR, 04].**

2. Granularité large : correspondant à un vocabulaire moins détaillé. Les ontologies de haut niveau ont une granularité large, du fait que les notions sur lesquelles elles portent peuvent être raffinées par des notions plus spécifiques **[FUR, 04].** Et c'est dans cette granularité que nous pouvons classer notre ontologie.

I.5 Cycle de vie d'une ontologie :

Un cycle de vie inspiré du génie logiciel qui se rapproche de cycle de vie prototypage, est proposé dans **[DIE, 01] ;** il comprend une étape initiale de détection et de spécification des besoins qui permet de circonscrire précisément le domaine des connaissances, une étape de conception, une étape de déploiement et de diffusion, une étape d'utilisation, une étape incontournable d'évaluation, et enfin une sixième étape consacrée à l'évolution et à la maintenance du modèle.

Trois étapes composent la phase de construction : conception, ontologisation, et opérationnalisation.

I.5.1 Détection et spécification des besoins :

Avant de démarrer le processus de construction de l'ontologie, il convient de bien définir le but visé. De ce fait, il est indispensable de bien délimiter l'objectif opérationnel de l'ontologie, particulièrement à travers des scénarios d'usage, sans oublier de délimiter le domaine de connaissance aussi précisément que possible.

I.5.2 La conceptualisation (normalisation, conception) :

Cette phase consiste à identifier dans un corpus (constitué à partir d'interviews d'experts du domaine, des documentations techniques existantes…) les connaissances du domaine puis trier ces connaissances.

La première tâche à effectuer est fastidieuse est ardue, vus qu'elle peut consister à rassembler les connaissances issues des interviews des experts du domaine, et à l'analyse de la documentation technique existante. Suite à cela, les interviews très ouvertes et les brainstormings doivent céder la place aux questionnaires permettant d'apporter plus de raffinements à un concept mis en évidence précédemment tel que préciser sa sémantique

différentielle. En plus de cela, faire une analyse des textes permet de détecter les termes et structures sémantiques (définitions, règles) présents dans le corpus.

I.5.3 L'ontologisation :

Il s'agit de modéliser et formaliser la base de connaissances filtrées issue de l'étape précédente et d'entamer la construction proprement dite de l'ontologie. Pour bien mener le processus de l'ontologisation, les axiomes doivent être cohérents, les définitions doivent être claires et objectives et indépendantes de tout choix d'implémentation, l'ontologie devrait être extensible sans modification. A ce niveau du cycle de vie de construction, l'ingénieur de la connaissance ainsi que l'expert du modèle formel de représentation de l'ontologie, assisté de l'expert du domaine doivent formaliser, autant que possible, le modèle conceptuel obtenu à l'étape précédente. Une part de connaissances du domaine peut, à ce niveau, être abandonnée du fait de l'impossibilité de lever certaines ambiguïtés ou du fait des limitations de l'expressivité du langage de représentation d'ontologie utilisé.

I.5.4 L'opérationnalisation :

Il s'agit d'effectuer une transcription de l'ontologie dans un langage formel et opérationnel de représentation de connaissances. Ce travail doit être mené par l'ingénieur de la connaissance.

I.5.5 L'évaluation et l'évolution d'une ontologie :

Deux niveaux peuvent être distingués dans l'évaluation d'une ontologie **[BOUG, 07]**

La vérification des propriétés formelles que l'ontologie ne devrait pas violer, sous peine de perdre son expressivité. L'ontologie doit être conforme à un modèle formel de représentation de connaissances. La vérification correspond à l'exigence « *building the system right* »

La validation, qui consiste à s'assurer de la conformité et fidélité sémantique de l'ontologie à un domaine de connaissance, c'est-à-dire que la sémantique exprimée dans l'ontologie doit être celle du domaine considéré. La validation correspond à l'exigence « *building the right system* ». De plus l'évaluation de l'ontologie en amont de son opérationnalisation est souhaitable pour éviter de propager des erreurs. Le test de la cohérence d'une ontologie peut nécessiter des déductions pour mettre en évidence des contradictions logiques entre axiomes. Cependant la validation des hiérarchies de concepts et/ou des relations doit être testée dès la phase d'ontologisation, aussi bien du point de vue formel que du point de vue sémantique **[BOUG, 07]**.

I.6 Langage et outils pour travailler avec les ontologies :

I.6.1 Langages:

Une des principales décisions à prendre dans le procédé de développement d'ontologies consiste à choisir le langage (ou l'ensemble de langages) dans lequel l'ontologie sera exprimée et utilisée. Parmi les langages développés pour les ontologies et les plus fréquemment utilisés, certains sont basés sur la syntaxe de XML, tels que XOL (Ontology Exchange Language), OML (Ontology Markup Language), RDF(Resource Description Framework) et RDF Schéma, Trois autres langages sont établis sur RDF(S) pour améliorer ses caractéristiques: OIL (Ontology Inference Layer), DAML+OIL et OWL (Web Ontology Language) qui est une révision de DAML+OIL qui utilise la conception et l'application de DAML+OIL et qui tend à s'imposer, et dans lequel nous avons exprimé notre ontologie puisque il s'avers le meilleur.

OWL : (Ontology Web Language) a été conçu pour être utilisé par les applications qui traitent le contenu de l'information. Il facilite grandement l'interopérabilité en fournissant plus de vocabulaire pour décrire les classes et les propriétés comme des approches orientées objets. Par exemple : les relations entre les classes, les cardinalités, l'égalité, le typage plus riche des propriétés, les caractéristiques des propriétés, etc. OWL a été conçu pour satisfaire le besoin d'un langage d'ontologie du web. Il ajoute plus de vocabulaire pour décrire les propriétés et les classes. On peut citer entre autre: les relations entre classes (par exemple la disjonction), les cardinalités (par exemple exactement un), l'égalité, typage plus riche des propriétés, caractéristiques des propriétés (par exemple la symétrie) et les classes énumérées. Bien que OWL soit dérivé de DAML+OIL qui est équivalent à un langage très expressif de la logique des descriptions, il n'est en tant que tel, équivalent à aucune logique de description. Certaines caractéristiques font qu'il n'existe aucun algorithme d'inférence décidable avec cette puissance d'expression. OWL fournit trois sous langages de plus en plus expressifs conçus pour l'usage des communautés spécifiques des utilisateurs et des développeurs: OWL Lite, OWL DL et OWL Full. **[01]**

Le langage OWL Lite convient aux utilisateurs qui ont principalement besoin d'une hiérarchie de classification et de contraintes simples. Par exemple, alors qu'il supporte des contraintes de cardinalité, il autorise seulement des valeurs de cardinalité 0 ou 1. Il devrait

être plus simple de fournir un outil d'aide pour OWL Lite que ses parents plus expressifs. OWL Lite fournit un chemin rapide de migration pour les taxonomies.

Le langage OWL DL correspond à une logique de description expressive. Il convient aux utilisateurs qui demandent un maximum d'expressivité tout en maintenant la complétude (garantie de calculer toutes les conclusions) et la décidabilité (tous les calculs doivent fournir en un temps fini). OWL DL inclut tous les constructeurs du langage OWL, mais ils peuvent être utilisés seulement sous certaines restrictions. OWL DL est appelé ainsi en raison de sa correspondance avec les logiques de description.

Le langage OWL Full convient aux utilisateurs qui demandent un maximum d'expressivité. Par exemple, une classe peut être traitée comme une collection d'individus et en même temps peut être vue comme un seul individu. OWL Full permet aussi à une ontologie d'augmenter le sens du vocabulaire prédéfini. Chacun de ces sous langages représente une extension par rapport à son prédécesseur plus simple, à la fois par ce qu'on peut exprimer légalement et aussi par ce qu'on peut conclure de manière valide **[ABE, 04].**

I.6.2 Outils :

Plusieurs outils de création et de manipulation d'ontologie qui existent qui sont téléchargeables et gratuits dont nous citons quelques un :

I.6.2.a Protégé : Protégé est le plus connu et le plus utilisé des éditeurs d'ontologie. Open source, développé par l'Université de Stanford, il a évolué depuis ses premières versions (Protégé'2000) pour intégrer à partir de 2003 les standards du web sémantique RDF (Resource Description Framework), RDFS (Resource Description Framework Schema) et notamment OWL. Il offre de nombreux composants optionnels: raisonneurs, interfaces graphiques. Dans Protégé 2000, le modèle de connaissance est basé sur une hiérarchie de classes explicite où l'ontologie est décrite de manière déclarative. Protégé est un éditeur d'ontologies pour les différents langages : RDF et OWL. Il dispose de Plugins pour ces deux langages.

I.6.2.b Swoop : est un éditeur d'ontologie développé par l'Université du Maryland dans le Cadre du projet MINDSWAP. Contrairement à Protégé, il a été développé de façon native sur les standards RDF et OWL, qu'il prend en charge dans leurs différentes

syntaxes (pas seulement XML). C'est une application plus légère que Protégé, moins évoluée en termes d'interface, mais qui intègre aussi des outils de raisonnement.

I.6.2.c OntoSaurus : [MAH & MIS, 07] Développé à l'Information Science Institut de l'Université de Southern California. Ontosaurus consiste en un serveur utilisant LOOM comme langage de représentation des connaissances, et en un serveur de navigation créant dynamiquement des pages HTML qui affichent la hiérarchie de l'ontologie; le serveur utilise des formulaires HTML pour permettre à l'usager d'éditer l'ontologie.

I.6.2.d Ontolingua : [MAH & MIS, 07] Développé à l'Université de Stanford, le serveur Ontolingua est le Plus connu des environnements de construction d'ontologies, il consiste en un ensemble d'outils et de services qui supportent la construction en coopération d'ontologies, entre des groupes séparés géographiquement.

I.6.2.e ODE : Développé au laboratoire d'Intelligence Artificielle de l'Université deMadrid.Les principaux avantages d'ODE (Ontology Design Environment) sont le module de conceptualisation pour construire des ontologies et le module pour construire des modèles.

II. Web sémantique :

Le Web constitue aujourd'hui une source d'informations majeure, dont la richesse est largement sous-exploitée. Il est constitué d'un gisement de documents, principalement textuels, encodés dans un langage particulier (HTML) permettant d'exprimer des liens entre un objet dans le document source (l'ancre) et un objet du document cible. Il est exploité par des dispositifs logiciels (navigateurs ou robots de recherche) qui traversent ces liens lorsqu'ils les rencontrent (ou lorsque l'utilisateur clique sur une ancre). Le travail d'exploitation du Web est donc principalement dévolu aux utilisateurs humains qui doivent analyser le contenu des pages pour déterminer sur quel lien cliquer. Des dispositifs logiciels peuvent les y aider, mais leur apport, bien que remarquable, reste limité car le contenu des documents du Web s'adresse aux utilisateurs humains.

La nouvelle génération du Web en l'occurrence le Web sémantique a pour ambition de lever cette difficulté. Les ressources du Web seront plus aisément accessibles aussi bien par l'homme que par la machine, grâce à la représentation sémantique de leurs contenus. Le principe du Web sémantique, comme celui du Web, est son ouverture : la possibilité de faire référence à des ressources distantes, non maîtrisées et la possibilité de les enrichir sans être entravé par des barrières physiques, techniques ou même réglementaires. Pour hériter de ces particularités du Web, le Web sémantique devra tirer parti d'un espace d'adressage global des ressources permettant d'y ajouter en permanence de l'information et d'y avoir accès sans entrave. Il devra offrir des langages permettant de partager et d'échanger la description de ces ressources. Dans le Web sémantique, les ressources seront ainsi accessibles et compréhensibles par des logiciels, qui pourront effectuer des raisonnements, des recherches plus « intelligentes » que les recherches par des mots clés comme aujourd'hui.

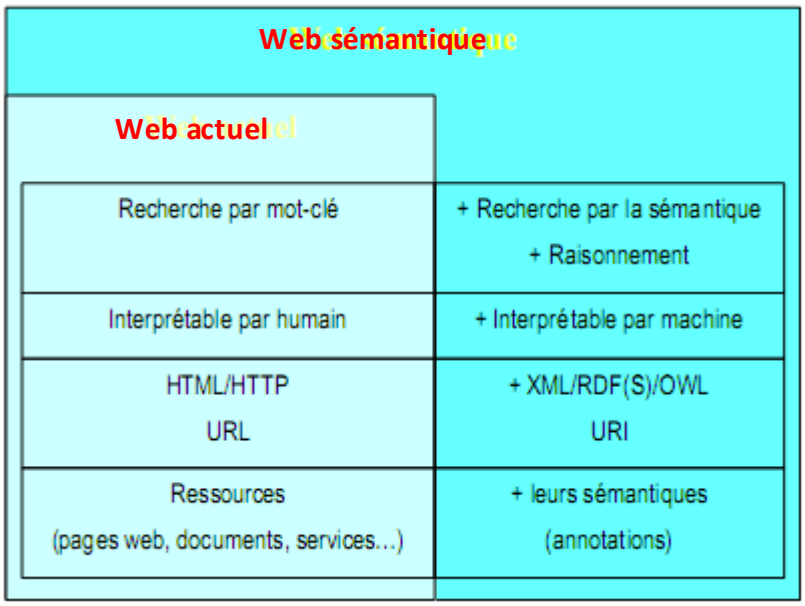


Figure1 : Le Web actuel et, le Web sémantique. **[BAC 06]**

II.1 Définitions :

L'expression **Web sémantique**, attribuée à Tim Berners-Lee au sein du W3C, fait d'abord référence à la vision du Web de demain comme un vaste espace d'échange de ressources entre êtres humains et machines permettant une exploitation, qualitativement supérieure, de grands volumes d'informations et de services variés. Espace virtuel, il devrait voir, à la différence de celui que nous connaissons aujourd'hui, les utilisateurs déchargés d'une bonne partie de leurs tâches de recherche, de construction et de combinaison des résultats, grâce aux capacités accrues des machines à accéder aux contenus des ressources et à effectuer des raisonnements sur ceux-ci. **[LAU 02]**

Le Web sémantique désigne un ensemble de technologies visant à rendre le contenu des ressources du World Wilde Web accessible et utilisable par les programmes et agents logiciels, grâce à un système de métadonnées formelles, utilisant notamment la famille de langages développés par le W3C **[01].**

.W3C : Le World Wide Web Consortium, abrégé par le sigle W3C, est un organisme de standardisation à but non-lucratif, fondé par Tim Berners-Lee en octobre 1994 comme un consortium chargé de promouvoir la compatibilité des technologies du World Wide Web telles que HTML, XHTML, XML, RDF, CSS, PNG, SVG et SOAP. Il a été fondé au MIT/LCS

(Massachusetts Institute of Technology / Laboratory for Computer Science) avec le soutien de l'organisme de défense américain DARPA - pionnier de l'Internet - et la Commission européenne.

.Métadonnée : Une métadonnée (mot composé du préfixe grec méta, indiquant l'auto-référence ; le mot signifie donc proprement «donnée à propos de donnée ») est une donnée servant à définir ou décrire une autre donnée quel que soit son support (papier ou électronique).

II.2 Architecture du web sémantique :

Dans cette section, nous présentons brièvement les composants de base sur lesquels on construit le Web sémantique, comme les montres la figure suivante :

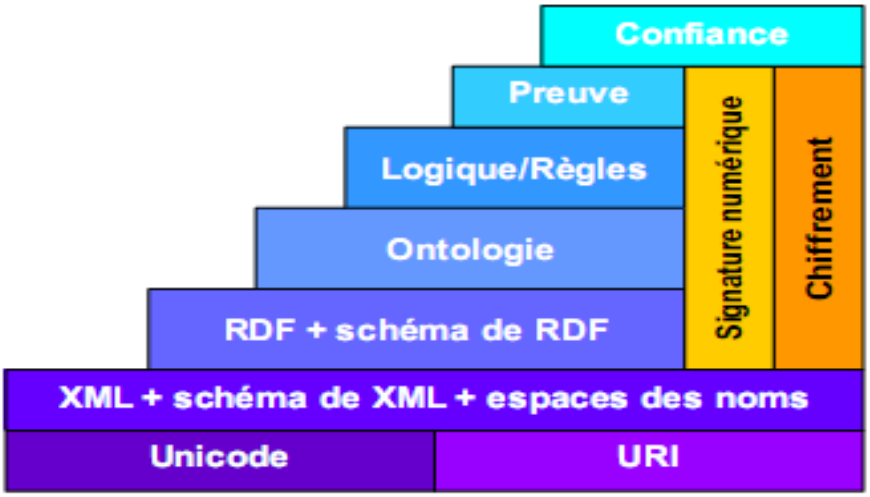


Figure 2 : l'architecture du web sémantique

1. URI : (Uniform Resource Identifier - identifiant uniforme de ressource). Est employée pour désigner le nom et/ou l'adresse d'une ressource dans le Web sémantique sans distinction d'une ressource physique ou une ressource abstraite. L'URI est une chaîne courte des caractères mais il n'est employé que pour référer des ressources par leur localisation. Notons aussi que quelque chose qui peut être identifié avec une URI peut être décrit, ainsi le Web sémantique peut raisonner au sujet des personnes, des endroits, des idées… Les URIs sont utilisés pour identifier (nommer) des ressources, mais sans procédé pour les récupérer.

Quelques exemples d'URI :

http://www.inria.fr/acacia/index.htm (pour une page web)

http://www.inria.fr/acacia/OntologyMatching.pdf (pour un article)

rstp://www.video.com/france.rm (pour une vidéo)

2. XML : (Extensible Markup Language) fournit une syntaxe pour les documents structurés (dans un structure hiérarchique, un arbre), mais n'impose aucune contrainte sémantique à la signification de ces documents. Un langage à balises combine le texte et les informations supplémentaires (méta-données) sur le texte. Les informations supplémentaires, telles que la structure, la police, la couleur… des textes, sont exprimées en utilisant des balises, qui sont mélangées avec le texte à présenter. En plus, le langage XML permet aux utilisateurs de définir eux-mêmes leurs balises.

3. Schéma XML : Un schéma XML est une description du type d'un document XML qui contient un ensemble de règles (contraintes sur la structure et le contenu du document) auxquelles un document XML doit se conformer afin d'être considéré valide selon ce schéma. DTD (Document Type Definition), RELAX NG (REgular LAnguage for XML Next Generation), XML Schema sont spécifiquement développés pour exprimer des schémas de XML.

4. Espace de nom : (namespace) est un contexte ou un conteneur abstrait contenant des noms, des termes, des mots qui représentent des objets, des concepts dans le monde réel. Un nom défini dans un espace de noms correspond à un et seulement un objet, deux objets ou concepts différents sont référencés par deux noms différents dans un même espace de noms.

Voici un exemple du schéma de XML pour décrire un article :

```
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="article">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="title" type="xs:string"/>
```

```
   <xs:element name="numOfPages" type="xs:integer"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

Et L'exemple de document XML ci-dessous se conforme à ce schéma :

```
<article
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="article.xsd">
 <title>Ontology base de données relationnelle</title>
 <numOfPages>15</numOfPages>
</article>
```

Et xs, xmlns, xsi sont des espaces des noms.

5. RDF : (Resource Description Framework) est un modèle de méta-données pour référencer des objets (ressources) et comment ils sont reliés l'un à l'autre. Dans ce modèle, les ressources sont identifiées (référencées) par les URIs et nous pouvons faire des déclarations à propos de ces ressources en employant des expressions sous forme «sujet – prédicat – objet », appelées des triplets. Le sujet est la ressource, la « chose » à décrire. Le prédicat est un attribut ou un aspect de cette ressource, et exprime un lien ou une relation entre le sujet et l'objet. L'objet est l'objet de la relation ou de la valeur de ce prédicat. La figure suivante montre un exemple d'un modèle RDF :

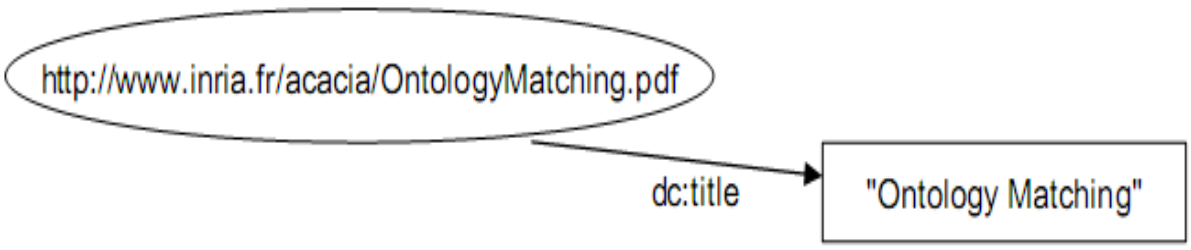


Figure3 : Un exemple de schéma RDF.

Sa représentation sous forme de triplet est :

<http://www.inria.fr/acacia/OntologyMatching.pdf> <dc:title> "Ontology Matching".

6. RDF Schéma : RDF Schema est un langage pour décrire des vocabulaires, des propriétés et des classes de ressources dans le modèle RDF. RDF Schema est une extension sémantique de RDF. Il fournit des mécanismes pour décrire des groupes de ressources similaires (classes) et des relations entre ces ressources (propriétés).
7. OWL : OWL (Web Ontology Language) est un langage pour représenter des ontologies dans le Web sémantique.

Dans l'architecture du Web sémantique proposée par Berners-Lee dans **[BER, 01]** nous voyons les liens entre des éléments du Web sémantique présentés ci-dessus. L'URI, l'Unicode, XML sont considérés comme des briques de base, sur lesquelles reposent des langages tels que RDF(S) permettant de décrire des « choses » (objets, concepts…) et leurs types ; puis des langages tels que OWL pour exprimer des relations entre des types des choses mais sans spécifier comment exploiter ces informations dans les calculs. En combinant avec la logique, les règles, des assertions autour du Web (stockées par des langages des couches inférieures tels que RDF(S), OWL) peuvent être employées pour déduire de nouvelles connaissances. Au-dessus de la couche de logique, c'est la couche de preuve. Ici, nous avons un langage universel pour représenter des preuves, qui permet à différents systèmes de partager leurs connaissances « originales » ou déduites. La provenance (l'origine) des connaissances, des données, des ontologies ou des déductions est authentifiée et assurée par des signatures numériques, dans le cas où la sécurité est importante ou le secret nécessaire, le chiffrement est employé. Enfin, le but final vers lequel nous nous orientons est la confiance, un Web sémantique et fiable, où nous pouvons effectuer plusieurs tâches complexes en sûreté.

III. E-Learning :

Plusieurs termes sont utilisés pour traduire le terme Anglais **E-Learning**. Les termes corrects sont **apprentissage en ligne**, **apprentissage virtuel** et **apprentissage électronique**. Parfois certains emploient le terme **E-formation**, un calque de l'anglais.

- "Le **e-Learning** est un processus d'apprentissage à distance s'appuyant sur des ressources multimédias, qui permet à une ou plusieurs personnes de se former à partir de leur ordinateur. Les supports multimédias utilisés peuvent combiner du texte, des graphismes en 2 ou 3 dimenssions, du son, de l'image, de l'animation et même de la vidéo."

- "Mode d'apprentissage basé sur l'utilisation des nouvelles technologies, permettant l'accès à des formations en ligne, interactives et parfois personnalisées, diffusées par l'intermédiaire d'internet, d'un intranet ou autre média électronique, afin de développer les compétences, tout en rendant le processus d'apprentissage indépendant de l'heure et de l'endroit."

Aujourd'hui, le e-Learning est lié spécialement à deux secteurs :
Des universités qui offrent des alternatives de formation à distance.
Des grandes entreprises qui l'utilisent avec le KM (Knowledge Management ou management des connaissances) pour former et actualiser les connaissances de ses employés de manière plus rapide.

. L'apprentissage est indépendant du temps et du lieu ;
.Permet une réduction des coûts (déplacement, hébergement…) ;
.C'est un moyen trop aisé pour des personnes qui n'ont pas la possibilité de se déplacer facilement ;
.L'acquisition des connaissances est permise pour n'importe qu'elle personne quelque soit son âge ou son niveau (ses compétences techniques) à condition quelle possède les outils de cette technologie ;
.Chacun devient responsable de son apprentissage (personnaliser la formation) ;
.Avoir un nombre non limité d'apprenants ;
.Une meilleure assimilation des connaissances (le suivi de l'apprenant est personnalisé) ;
.Elargir les conditions d'accès des publics à la formation ; **[KMS, 04]**

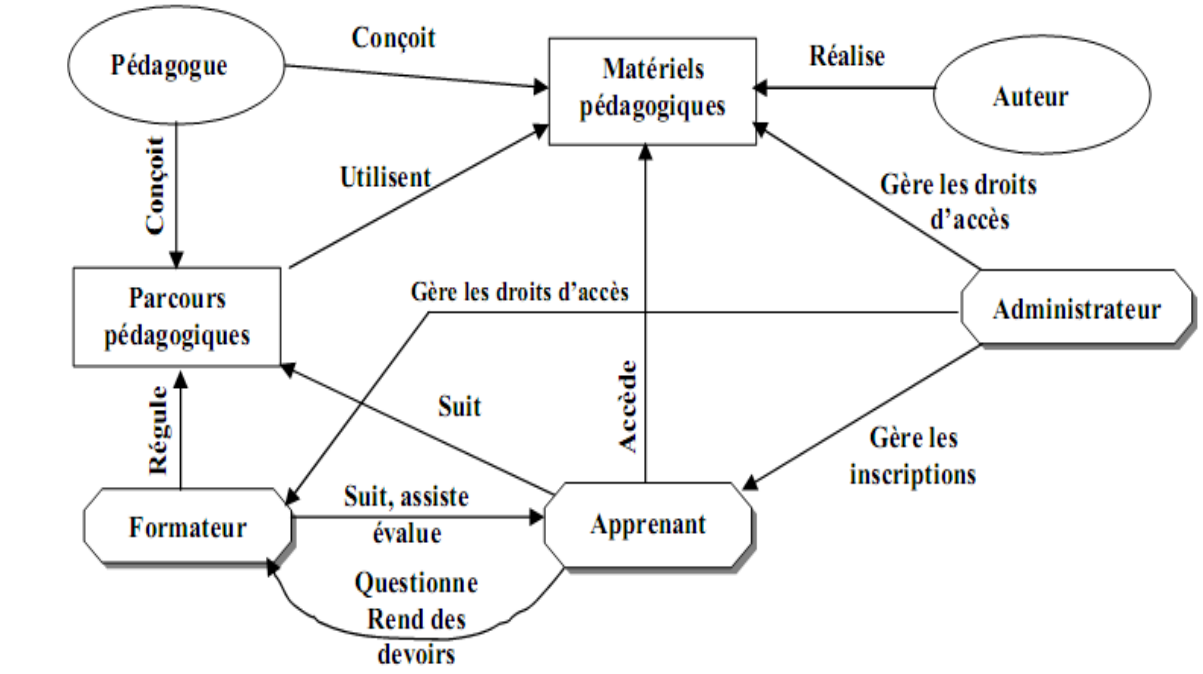
III.1 Les plates formes E-Learning :

La plate-forme est un outil de diffusion et de gestion des connaissances, associant des contenus de cours à des moyens de communication, des outils d'entraînement et d'évaluation. Les plateformes sont très nombreuses et offrent un choix très varié de services pour les usagers. Parmi les acteurs d'une plate forme : le formateur, l'apprenant, et l'administrateur qui est le responsable de la plate-forme de e-Learning. Il gère les comptes utilisateurs, les groupes, les connaissances du domaine d'enseignement et leurs mises à jour.

La plate-forme de e-Learning est l'élément central d'un système de formation à distance. Ses principales fonctions sont :

- Diffusion du contenu pédagogique.
- La construction des parcours de formation personnalisés.
- La gestion des évaluations des apprenants.
- Le suivie et l'assistance individuelle des enseignants aux apprenants dans leurs progressions ;
- La gestion administrative des apprenants.

La figure suivante illustre un schéma général des plates formes pour la formation à distance.



: Accès en lecture et en écriture.

: Accès en lecture.

: Ressources.

Figure 4: Schéma général des plates formes pour la formation à distance **[SEB, 01]**

- **l'enseignant pédagogue** : chargé de concevoir le matériel pédagogique des cours et crée les parcours pédagogiques de son enseignement.
- **L'auteur (concepteur informatique)** : ou l'enseignant lui-même, réalise les médias (texte, image, etc.).
- **Le formateur (tuteur)** : effectue un suivi du travail des apprenants (évaluation, temps de parcours, etc.), ainsi qu'une assistance dans l'apprentissage de ces derniers. Il peut éventuellement réguler leurs parcours pédagogiques.
- **L'apprenant** : consulte en ligne ou télécharge les contenus qui lui sont recommandés, organise son travail, effectue des exercices, s'auto-évalue et transmet des travaux au tuteur qui les évalue.
- **L'administrateur** : installe et assure la maintenance du système, s'occupe de l'inscription des apprenants, gère les accès et les droits aux ressources pédagogiques.

L'enseignant et l'apprenant communiquent individuellement ou en groupe, créent des thèmes de discussion et collaborent pour produire des documents communs.

IV. Apport du web sémantique au domaine du e-learning: [HAC, 09]

Une application e-learning exige un certain nombre de défis concernant le processus d'apprentissage : rapide, juste à temps, pertinent et moins cher. Les propriétés clés de l'architecture du Web Sémantique (sens partagé commun, ontologies, méta-données traitables par les machines), offertes par un ensemble adéquat d'agents, apparaissent suffisamment puissantes pour satisfaire les exigences du e-learning.

Le Web sémantique permet le développement des ontologies et l'annotation des ressources d'apprentissage, son utilisation devient très utile et adéquate pour implémenter un système e-learning.

Pour la réalisation des exigences du e-learning nous pouvons bénéficier des avantages du Web sémantique. Le tableau suivant, traduit les travaux de **[STO, 01],** montre cet apport :

Exigences	**e-learning**	**Web Sémantique**
Livraison (Delivery)	Traction-l'étudiant détermine l'ordre du jour.	Les supports d'apprentissage (éléments de connaissances) sont distribués sur le web, mais ils sont généralement indexés par des ontologies communes. Cela permet la construction d'un cours spécifique à l'utilisateur, en utilisant des requêtes sémantiques sur les sujets qui il l'intéresse.
Capacité à Répondre (Responsiveness	Réactionnaire - répond au problème actuel.	Les agents logiciels sur le web sémantique peuvent utiliser un langage de service commun, qui permet la coordination entre les agents et la livraison proactive de supports d'apprentissage dans le contexte des problèmes réels. L'idée est que chaque utilisateur ait son propre agent personnalisé qui communique avec d'autres agents.

Accès (Access)	Non linéaire - permet l'accès direct aux connaissances dans n'importe quel ordre donnant un sens à la situation actuelle.	L'utilisateur peut décrire la situation actuelle (but de l'apprentissage, connaissances précédentes...) et exécuter des requêtes sémantiques pour le support d'apprentissage approprié. Le profil d'utilisateur est également pris-en considération. L'accès aux connaissances peut être augmenté par une navigation sémantiquement définie.
Symétrie (Symmetry)	Symétrique –l'apprentissage apparait comme une activité intégrée	Le Web Sémantique offre le potentiel de devenir une plate-forme intégrée pour tous les processus économiques dans une organisation, y compris les activités d'apprentissage.
Modalité (Modality)	Continu – l'apprentissage se déroule en parallèle avec les tâches professionnelles et ne s'arrête jamais.	La livraison active d'information (basée sur les agents personnalisés) crée un environnement d'apprentissage dynamique qui est intégré dans les processus économique.
Autorité (Authority)	Distribué- le contenu se construit à partir des interactions des participants et des enseignants.	Le Web Sémantique doit être décentralisé le plus possible. Cela permet une efficace gestion coopérative du contenu.
Personnalisation (Personalization)	Personnalisé- le contenu est déterminé en fonction des besoins individuels de l'utilisateur et vise à satisfaire les besoins de chaque utilisateur.	L'utilisateur, en utilisant son agent personnalisé, recherche le support d'apprentissage adapté à ses besoins. L'ontologie est le lien entre les besoins d'utilisateur et les caractéristiques du support d'apprentissage.
Adaptativité (Adaptivity)	Dynamique - le contenu change constamment à travers les entrées de l'utilisateur : ses expériences, ses nouvelles pratiques, ses règles de travail et l'heuristique, etc.	Le Web Sémantique permet l'utilisation de connaissances distribuées délivrées sous diverses formes, avec des annotations sémantiques du contenu. La nature distribuée du Web Sémantique permet une amélioration continue des supports d'apprentissage.

Conclusion :

Dans ce premier chapitre nous avons vue quelques généralités à propos du web sémantique des ontologies et du e-Learning ainsi leurs interactions. Pour le second chapitre nous allons présenter l'ontologie de notre application suivie de son édition avec l'outil Protégé.

CHAPITRE II

Construction de l'ontologie.

Introduction :

Apres avoir donné quelques généralités sur les ontologies, le web sémantique et le E-Learning dans le premier chapitre, nous allons consacrer les chapitres qui reste a la réalisation de notre projet qui sera débuter dans ce deuxième chapitre ou nous allons éditer notre ontologie.

En premier lieu nous parlerons des transformations du modèle UML vers le code OWL, en deuxièmes lieu nous allons passer à l'édition de notre ontologie ensuite sa visualisation et ça sous l'outil Protégé.

VI. Les règles de passage du modèle UML vers l'ontologie OWL :

Notre point de départ pour la construction de la partie déclarative de l'ontologie est le diagramme de classes UML. Nous optons alors la transformation de ce modèle semi formel en une ontologie formalisée en OWL.

I.1 Le processus de transformation :

Dans le cas qui nous occupe, nous sommes intéressés par la conception d'un processus de transformation qui permet de formaliser des connaissances exprimées de façon semi-formelle en une ontologie OWL formelle, notons que le modèle UML est similaire a l'ontologie OWL, pour cela on a suivi le processus de transformation parallèle selon ACM1. Nous présentons ici quelques règles de transformation, suivi par le détail du processus de transformation tel que réalisé.

UML	OWL
Class	owl :Class
Generalization	rdfs :subclassOf.
Association	owl :ObjectProperty ou owl :Class.
Attributes	owl :DatatypeProperty
InstanceOf	rdf :type
Multiplicity	owl:minCardinality,owl :maxCardinality

Tableau 1.Les règles de transformation **[BEN, 09]**

1 : Association for Computing Machinery

I.2 Les différentes classes de modèle UML :

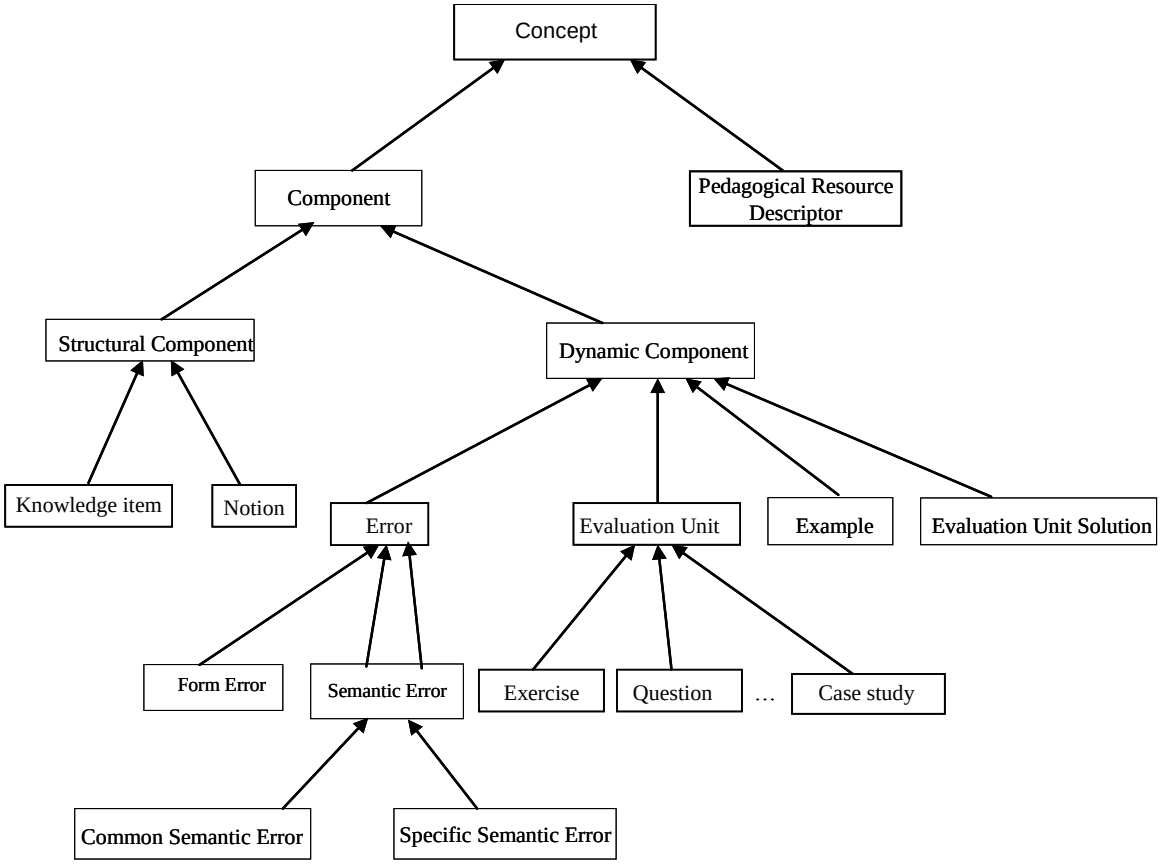


Figure5 : la hiérarchie des classes correspondantes à notre ontologie **[BOU, 10]**

I.3 Génération de l'ontologie OWL correspondant au modèle UML :

Donnant quelques exemples pour la correspondance des classes et des relations :

I.3.a Les classes :

Les classes	leurs correspondances en OWL.
Component	<owl:Class rdf:ID=" Component ">
Dynamic_component	<owl:Class rdf:ID=" Dynamic_component ">
Error	<owl:Class rdf:ID=" Error ">
Evaluation_unit	<owl:Class rdf:ID=" Evaluation_unit">
Exercice	<owl:Class rdf:ID=" Exercice">

Qestion	<owl:Class rdf:ID=" Qestion">
Evaluation_unit_solution	<owl:Classrdf:ID=Evaluation_unit_solution">
Example	<owl:Class rdf:ID=" Example">
Structural_component	<owl:Class rdf:ID=" Structural_componen
Knowledge_item	<owl:Class rdf:ID=" Knowledge_item ">
Notion	<owl:Class rdf:ID=" Notion ">

Tableau 2. La correspondance des classes UML en OWL.

I.3.b Les relations :

Les relations	**leurs correspondances en OWL.**
Annotates	<owl:ObjectProperty rdf:ID=" Annotates "/>
Required_for	<owl:ObjectProperty rdf:ID=" Required_for "/>
Is_examples_of	<owl:ObjectProperty rdf:ID=" Is_examples_of"/>
Has_examples	<owl:ObjectProperty rdf:ID=" Has_examples "/>
Is_related_to	<owl:ObjectProperty rdf:ID=" Is_related_to "/>
Has_potential_error	<owl:ObjectPropertyrdf:ID="Has_potential_error "/>
Evaluated_by	<owl:ObjectProperty rdf:ID=" Evaluated_by "/>
Caracteristic	<owl:Class rdf:ID=" Caracteristic ">
Semantic annotation	<owl:Class rdf:ID=" Semantic annotation ">

Tableau 3. La correspondance des relations UML en OWL

VII. les règles d'inférences associées à l'ontologie :

Les règles d'inférences n'ont pas de correspondance en OWL, pour cela on a utilisé le langage SWRL. Les règles d'inférence associées à notre ontologie sont décrites c'est dessous :

1. Rule-T1*: **if** n1 N-Composed-of-N n2 **and** n2 N-Composed-of-N n3 **then** n1 N-Composed-of-N n3*

2. Rule-T2*: **if** c1 Is A Structural Component **then** c1 Is A Component*

3. Rule-T3*: **if** n1 Is Required for n2 **and** n2 Is Required for n3 **then** n1 Is Required for n3*

4. Rule-T4: ***if** n1 N-Composed-of-N n2 **and** n2 N-Composed-of-KI n3 **then** n1 N-Composed-of-KI n3*

5. Rule-5*: **if** n N-Composed-of-KI k **and** k EvaluatedBy x **then** n1 EvaluatedBy x*

6. Rule-6*: **if** n N-Composed-of-KI k **and** k Has-Potential-Error e **then** n Has-Potential-Error e*

7. Rule-7*: **if** n1 N-Composed-of-N n2 **and** n3 Is Required for n2 **then** n3 Is Required for n1*

8. Rule-8*: **if** n N-Composed-of-KI k1 **and** k2 Is Required for k1 **then** k2 Is Required for n*

- La correspondance de ces règles en SWRL dans Protégé est respectivement décrite respectivement comme suite :

1. Rule-T1*: N-Composed-of-N* (?n1, ?n2)/*N-Composed-of-N* (?n2, ?n3)⟶*N-Composed-of-N* (?n1, ?n3)

2. Rule-T2 : *Structural Component* (?c1) ⟶ C*omponent (?c1)*

3. Rule-T3*: Required for* (?n1, ?n2) /\ *Required for*(?n2, ?n3) ⟶ *Required for*(?n1, ?n3)

4. Rule-T4: *N-Composed-of-N* (?n1, ?n2) /\ *N-Composed-of-KI* (?n2, ?n3) ⟶ *N-Composed-of-KI* (?n1, ?n3)

5. Rule-5*: N-Composed-of-KI* (?n, ?k) /\ *EvaluatedBy*(?k, ?x)⟶ *EvaluatedBy*(?n, ?x).

6. Rule-6*: N-Composed-of-KI* (?n, ?k) /\ *Has-Potential-Error* (?k, ?e) ⟶ *Has-Potential-Error* (?n, ?e).

7. Rule-7 : *N-Composed-of-N* (?n1, ?n2) /\ *Is Required for*(?n3, ?n2) ⟶ *Is Required for* (?n3, ?n1).

8. Rule-8*: N-Composed-of-KI*(?n, ?K1) /\ *Is Required for*(?K2, ?K1) ⟶ *Is Required for*(?K2, ?n).

VIII. Génération de la base de connaissance pour l'enseignement des bases de données relationnelles correspondante à notre ontologie : [RKA, 09]

III.1 Les différentes instances de l'ontologie :

III.1.a Les instances des notions et des items de connaissances :

Notions	Sous Notions	Items de connaissances
1. Introduction au modèle relationnel	1. Rappels sur les bases de données.	1. Principe de l'approche bases de données 2. Définition d'une base de données 3. Catégories des bases de données
	2. Historique du modèle relationnel.	4. Origine du modèle relationnel 5. Objectifs du modèle Relationnel
	3. La relation.	6. Définition d'une relation 7. Attribut d'une relation 8. Degré d'une relation 9. Clé d'une relation 10. Schéma d'une relation 11. Domaine d'un attribut 12. Valeur nulle d'un attribut 13. Occurrence d'une relation 14. Cardinalité d'une relation
	4. Les règles d'intégrité.	15. Contrainte d'unicité de la clé 16. Contrainte de référence 17. Contrainte d'entité 18. Contrainte de domaine
2. Algèbre Relationnelle	5. Les opérateurs unaires	19. Principe des opérateurs unaires 20. Opérateur de la sélection 21. Opérateur du complément 22. Opérateur de la projection 23. Opérateur de l'auto jointure 24. Syntaxe d'une opération algébrique unaire
	6. Les opérateurs binaires	25. Principe des opérateurs

		binaires 26. Opérateur de l'union 27. Opérateur de l'intersection 28. Opérateur de la différence 29. Opérateur du produit cartésien 30. Opérateur de thêta-produit 31. Opérateur de la jointure naturelle 32. Opérateur de la jointure extérieure 33. Opérateur de la semi-jointure 34. Syntaxe d'une opération algébrique binaire
	7. Opération algébrique	35. Syntaxe de la condition 36. Lexique du mot 37. Syntaxe d'une opération Algébrique
3. Arbre algébrique	8. La représentation graphique des operateurs unaire	38. Principe des symboles unaires. 39. Formalise de la sélection. 40. Formalise du complément. 41. Formalise de la projection. 42. formalise de renommage. 43. formalise de représentation des opérateurs unaires.
	9. La représentation graphique des operateurs binaire.	44. Principe des symboles binaires 45. Formalise d'union 46. Formalise d'intersection 47. Formalise de la différence 48. Formalise du produit cartésien 49. Formalise thêta-produit 50. Formalise de la jointure naturelle 51. Formalise de la jointure extérieure 52. Formalise de la semi jointure gauche 53. Formalise de la semi jointure droite 54. Formalise de représentation des opérateurs Binaires
4. Langage SQL	10. Définition de données avec SQL	55. Création d'une base de données relationnelle avec SQL

		56. Création d'une table avec SQL 57. Modification d'une table avec SQL 58. Renommer une table avec SQL 59. Suppression d'une table avec SQL
	11. Manipulation de données avec SQL	60. Expression insert avec SQL 61. Expression update avec SQL 62. Expression delete avec SQL 63. Expression de la projection avec SQL 64. Expression de la sélection avec SQL 65. Expression de complément avec SQL 66. Expression de l'intersection avec SQL 67. Expression de la différence avec SQL 68. Expression de la division avec SQL 69. Expression de l'union avec SQL 70. Expression du produit cartésien avec SQL 71. Expression de la jointure naturelle avec SQL 72. Expression du thêta-produit avec SQL 73. Auto jointure avec SQL 74. Expression de la jointure extérieure avec SQL 75. Expression de la semi-jointure avec SQL
	12. Les autres fonctionnalités du SQL	76. Recherche des valeurs nulles 77. Les opérateurs arithmétiques 78. La fonction MAX avec SQL 79. La fonction MIN avec SQL 80. La fonction COUNT avec SQL 81. La fonction SUM avec SQL 82. La fonction AVG avec SQL 83. La fonction DISTINCT avec SQL 84. La fonction Order By avec SQL 85. La fonction Group By avec SQL

		86. La fonction Between avec SQL 87. La fonction Like avec SQL
5. DFA Normalise	13. Les dépendances fonctionnelles et les trois formes normales.	88. Principe des dépendances fonctionnelles 89. Propriétés des dépendances fonctionnelles 90. Les dépendances fonctionnelles élémentaires 91. Les dépendances fonctionnelles directes 92. Graphe de dépendances fonctionnelles 93. La fermeture transitive 94. La couverture minimale 95. Principe de normalisation 96. La première forme normale 97. La deuxième forme normale 98. La troisième forme normale 99. La forme normale de Boyce_Codd
	14. Les dépendances multivaluées et La forme normale	100. Principes des dépendances multivaluées 101. Propriétés des dépendances multivaluées 102. Principe de normalisation 103. La quatrième forme normale
	15. Les dépendances de jointures et La cinquième forme normale	104. Principe de dépendances de jointures 105. Principe de normalisation 106. La cinquième forme normale
	16. Algorithmes de normalisation	107. Principe de normalisation 108. Algorithmes de normalisation par synthèse 109. Algorithmes de normalisation par Décomposition

Tableau4 : Les notions et ces items de connaissances

III.1.b Les instances des erreurs :

- *Les erreurs lexico-syntaxique :*

Num err	Libellé de l'erreur	Num

		ItemC
1	Mot trop long	36
2	Point virgule manquant	37
3	Nombre réel erroné (possédant un caractère différent de «.» et différent d'un chiffre)	36
4	Absence d un identificateur de relation au début de l'opération algébrique	37
5	Absence de séparateur :=avant l'opérateur algébrique	37
7	absence du premier identificateur de relation après la parenthèse ouvrante dans l'opération algébrique UNION	26
8	manque du séparateur virgule entre deux identificateurs dans une opération algébrique UNION	26
9	Absence du deuxième identificateur de relation après le séparateur virgule dans une opération algébrique binaire UNION	26
10	parenthèse fermante manquante dans l opération algébrique UNION	26
11	opérateur d''affectation mal écrit manque de '=' après les deux points	37
12	parenthèse ouvrante manquante après l'opérateur algébrique AUTOJOION	23
13	parenthèse fermante manquante dans l opération algébrique AUTOJOIN	23
14	parenthèse ouvrante manquante après l'opérateur algébrique SELECTION	20
15	absence de l'identificateur de relation après la parenthèse ouvrante dans l opération SELECTION	20
16	Manque de séparateur | entre l'identificateur de relation et la condition dans la sélection	20
17	parenthèse fermante manquante dans l'opération algébrique SELECTION	20
18	parenthèse ouvrante manquante après l'opérateur algébrique projection	22
19	absence de l'identificateur de relation après la parenthèse ouvrante dans l opération de projection	22
20	Manque de séparateur '|' entre l identificateur de relation et la liste d	22

	attributs dans la projection	
21	parenthèse fermante manquante dans l opération algébrique de projection	22
22	parenthèse ouvrante manquante après l opérateur algébrique THETAPRODUCT	30

- *Les erreurs sémantiques spécifiques :*

Num err	***Libellé de l'erreur***	***Num ItemC***
23	Opérateur algébrique du complément trouvé alors qu'il n'est pas prévu par l'exo	21
24	Opérateur algébrique de projection introuvable alors qu'il est prévu par l'exo	22
25	Opérateur algébrique de la semi jointure introuvable alors qu'il est prévu par l'exo	33
26	Opérateur algébrique d'intersection trouvé alors qu'il n'est pas prévu par l'exo	28
27	Opérateur algébrique de complément introuvable alors qu'il est prévu par l'exo	21
28	le deuxième identificateur d'attribut utilisé dans la projection n'est pas le bon	22
29	vous n'avez pas utilisez le bon identificateur de relation pour la projection	22
30	le nombre d'attributs dans la projection est différent de deux	22
31	le premier identificateur de relation dans l'union n'est pas le bon	26
32	le deuxième identificateur de relation dans l'union n'est pas le bon	26
33	le nombre d'attributs dans la projection est diffèrent de un	22
34	le premier identificateur de relation dans l'opération algébrique de différence n'est pas le bon	27
35	le deuxième identificateur de relation dans l'opération algébrique de différence n'est pas le bon	27

- *Les erreurs sémantiques communes :*

Num err	***Libellé de l'erreur***	***Num ItemC***

122	les deux relations utilisées dans la SEMI JOINTURE n'ont pas d'attributs communs	33
110	les deux relations dans l'opération algébrique d INTERSECTION n'ont pas le même schéma relationnel	27
107	les deux relations utilisées dans la jointure naturelles n'ont pas d'attributs communs	31
105	les deux relations dans l'opération algébrique d'UNION n'ont pas le même schéma relationnel	26
104	les deux relations utilisées dans la jointure extérieure n'ont pas d'attributs communs	32
110	les deux relations dans l'opération algébrique d INTERSECTION n'ont pas le même schéma relationnel	27
134	vous avez fait la différence entre deux relations identiques	27
130	vous avez fait l union entre deux relations identiques	26

III.1.c Les instances de la classe exercices sous classe de Evaluation unit et de leurs solution (Evaluation unit solution) :

Nous allons pour exemples trois exercices qui auront des relations avec certains items.

Exercices1 :

Ecrire un programme SQL qui permet de rajouter le produit peinture a la table produit qui a pour attributs : code produit et designation.

Solution1 :

INSERT INTO PRODUIT (CODE_PROD ,DESCRIPTION)

VALUES('P','Peinture') ;

Exercices2 :

Exprimer en algèbre relationnel, la requête suivante :<< la liste des noms et prénoms de maires dont le niveau d'étude est "universitaire" et l'âge supérieur à 40ans>>

Solution2 :

R1 : = SELECTION (Maire| niveau études = universitaire && âge > 40)
R2 : = PROJECTION (R1 | nom, prénom) --> Résultat

Exercices3 :

Une société de location de véhicules est organisée en filiales. Chaque filiale est chargée de l'entretien des véhicules qu'elle loue à ses clients. Chaque location donne lieu à un contrat précisant la date de début et la date de fin de location. Donner le modèle relationnel correspondant ?

Solution3 :

Filiale (num _filiale, nom filiale, adresse filiale)
Véhicule (num _véhicule, marque véhicule, puissance véhicule)
Client (num _client, nom client, prénom client, adresse client)
Louer (num _client, num véhicule, date début location, date fin location)

- Un exercice est utilisé pour contrôler l'assimilation par l'apprenant d'une ou plusieurs notions, il fait référence à un ensemble d'items de connaissances, la résolution d'exercices permet d'évaluer la capacité de l'apprenant à combiner l'utilisation de plusieurs notions (et/ou plusieurs items de connaissances) du domaine pour répondre au problème posé par l'exercice**.** Ces exercices font référence aux notions suivantes :

Exercice1 **: Notion :** 4. Langage SQL et 1. Introduction au modèle relationnel

Sous-notion : 1. Rappels sur les bases de données, **3**. La relation, **4**. Les règles d'intégrité, **11**. Manipulation de données avec SQL.

Item : 1, 2, 6, 7, 9, 15, 55, 56, 60.

Exercice2 **: Notion :** 1. Introduction au modèle relationnel et 2. Algèbre relationnelle

Sous-notion : 1. Rappels sur les bases de Données, 3. La relation, 4. Les règles d'intégrité, 5. Les opérateurs unaires **et** 6. Les opérateurs binaires

Item : 6, 7, 8, 9, 20, et 27

Exercice3 **: Notion :** 1. Introduction au modèle relationnel

Sous-notion : 1. Rappels sur les bases de données. 4. Les règles d'intégrité.

Item : 1, 2, 6, 7, 9, 10.

IX. Edition de l'ontologie avec l'outil Protégé :

Le diagramme UML de notre ontologie possède des cardinalités multiples qu'OWL Lite ne peut par prendre en considération.

Chaque classe de notre ontologie est un ensemble d'individus, et vue qu'OWL Full offre un plus haut niveau, car il représente les classes comme une collection d'individu et en même temps comme un seul individu. Donc l'OWL Dl est le langage adéquat pour notre ontologie.

Outre, notre ontologie possède des règles d'inférences que nous allons représenter avec SWRL car le OWL DL offre des balaise pour le langage SWRL.

Nous avons choisit protégé comme outil d'édition de notre ontologie car il permet de représenter tous les concepts de l'ontologie sous forme OWL DL, il permet aussi la création des règles d'inférences avec la syntaxe SWRL. Avec protégé on peu aussi générer notre ontologie sous forme d'un fichier OWL en lui intégrant les règles d'inférences SWRL.

Grâce à l'interface graphique intermédiaire, on n'a pas besoin d'exprimer ce que l'on a à spécifier dans un langage formel. Il suffit juste de remplir les différents formulaires correspondant à ce que l'on veut spécifier (classes, attributs, propriétés...).

IV.1 La création des classes :

Tout d'abord il faut être dans un nouveau projet cliquez sur l'onglet OWLClasse. La partie de gauche permet la création des classes, celle de droite permet sa modification.

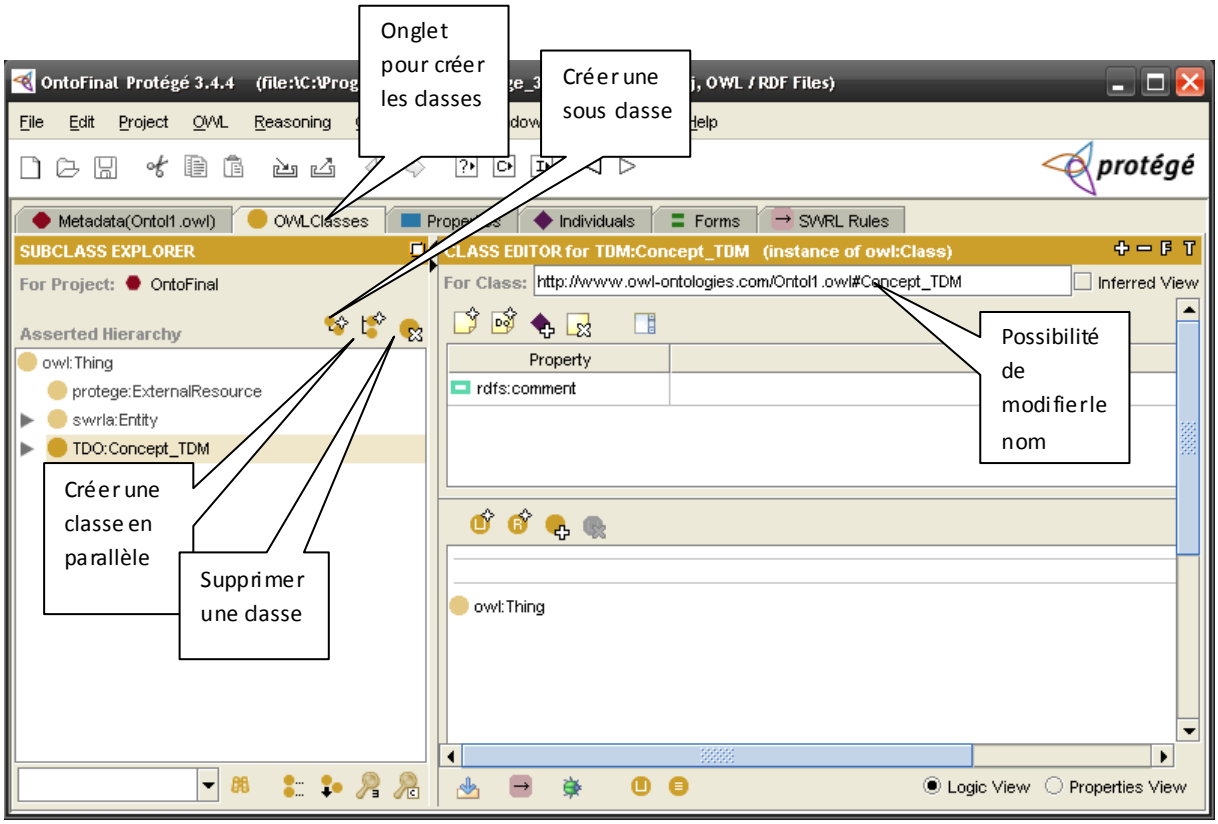


Figure6 : fenêtre pour la création des classes.

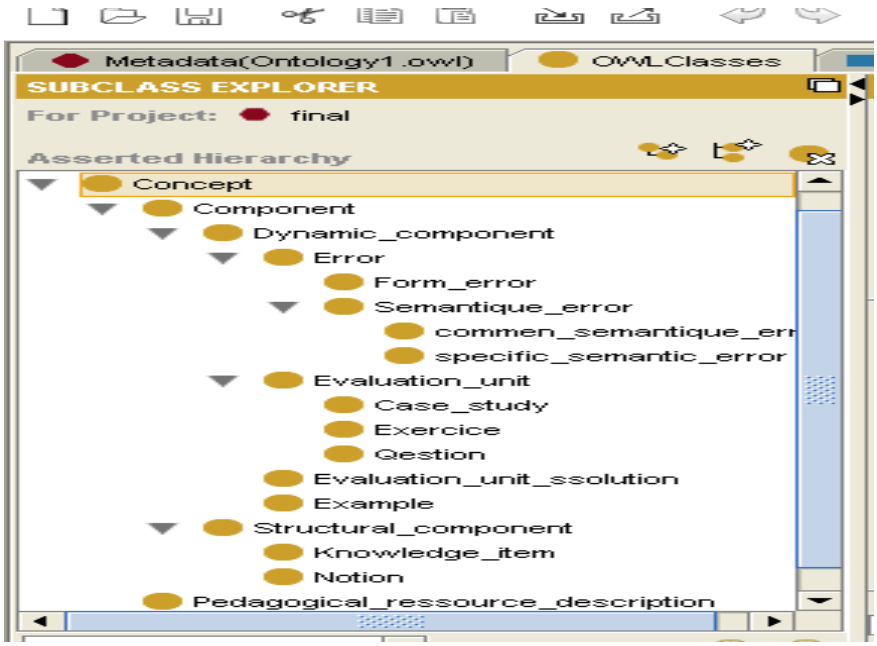


Figure7 : l'ensemble des classes de l'ontologie.

IV.2 Les relations :

Protégé permet aussi la création des relations entre les classes à laide de l'onglet Propriétés **figure3.**

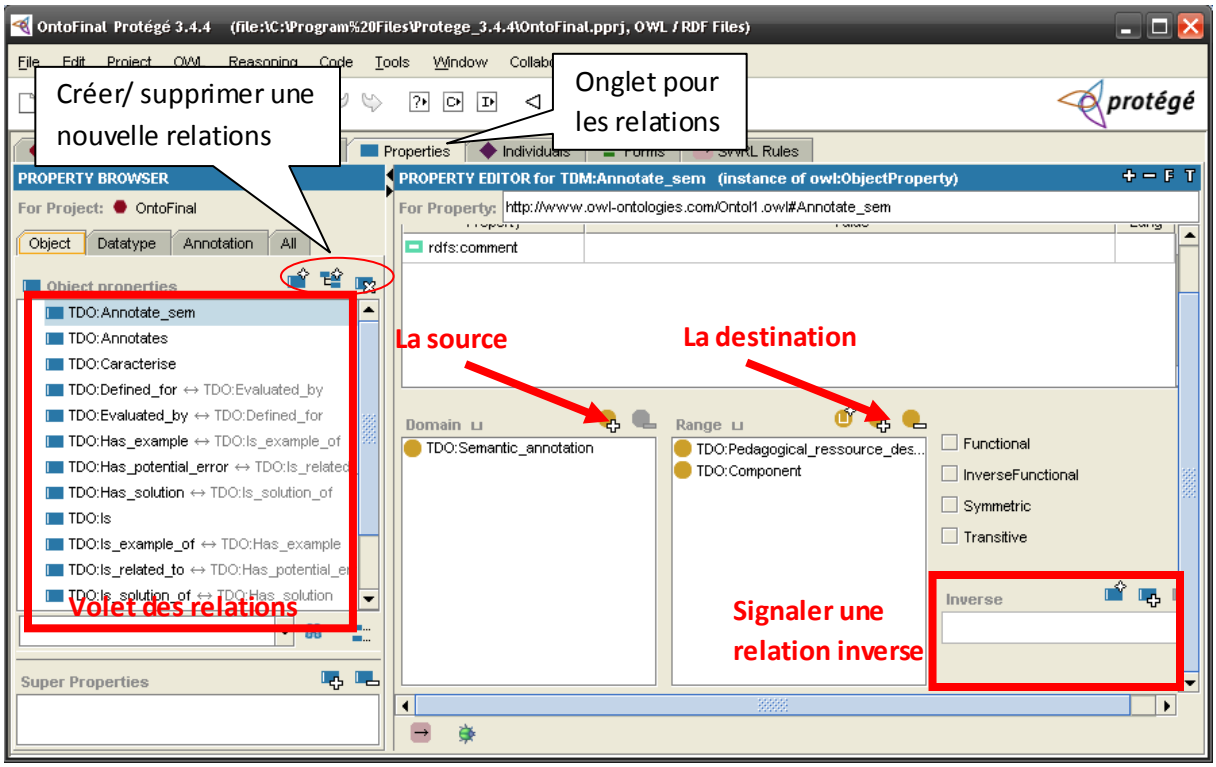


Figure8 : Création des relations avec Protégé.

Et comme le lien de composition appartenant à la sémantique de l'UML, n'a pas de représentation directe en OWL.

La solution choisie est de créer la propriété *N-Composed-of-N entre deux notion et la propriété N-Composed-of-KI* entre notion et items de connaissance au concept source et au concept destinataire désigné par le lien de composition

IV.3 Les individus :

L'insertion des individus dans différentes classes est expliquée dans la figure suivante :

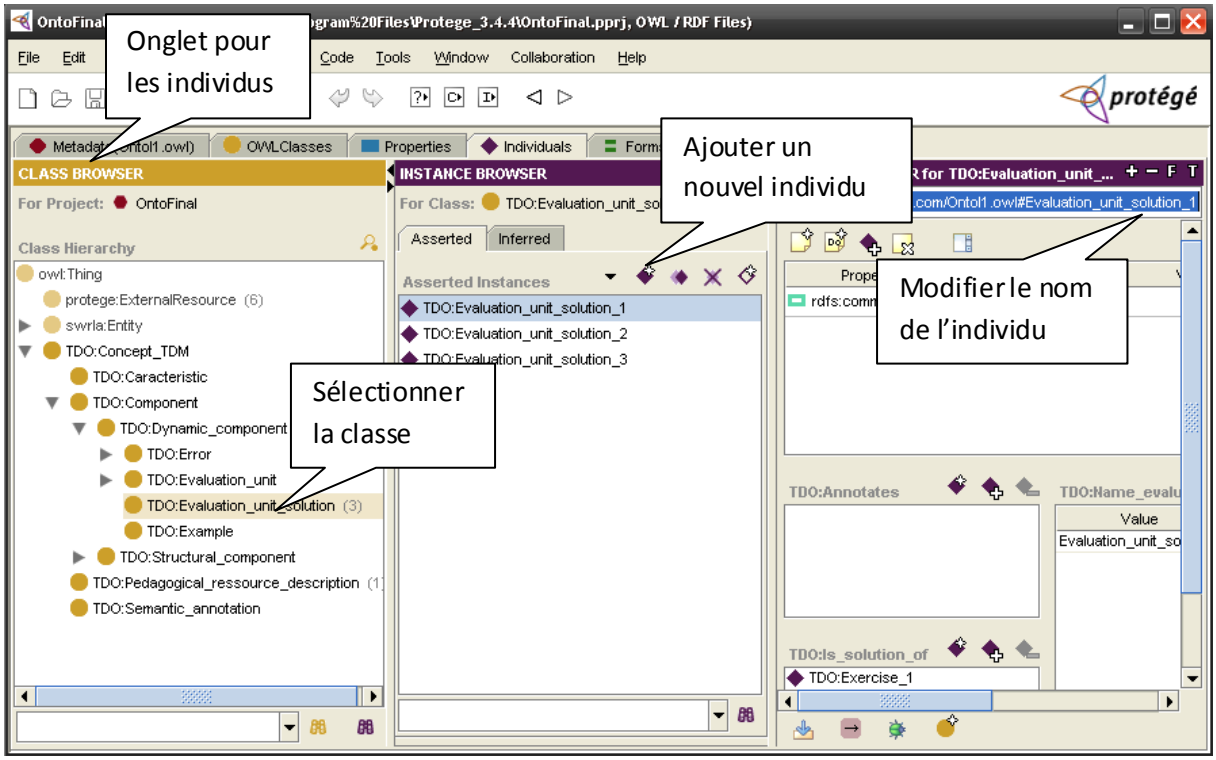


***Figure9** : insertion des individus dans leurs classes*

IV.4 Les règle d'inférences :

Pour introduire les règles d'inférences allez à l'onglet SWRL Rules (l'obtenir à partir Reasoning/ open SWRL Rule) ensuite ajouter les règles et introduire le code correctement.

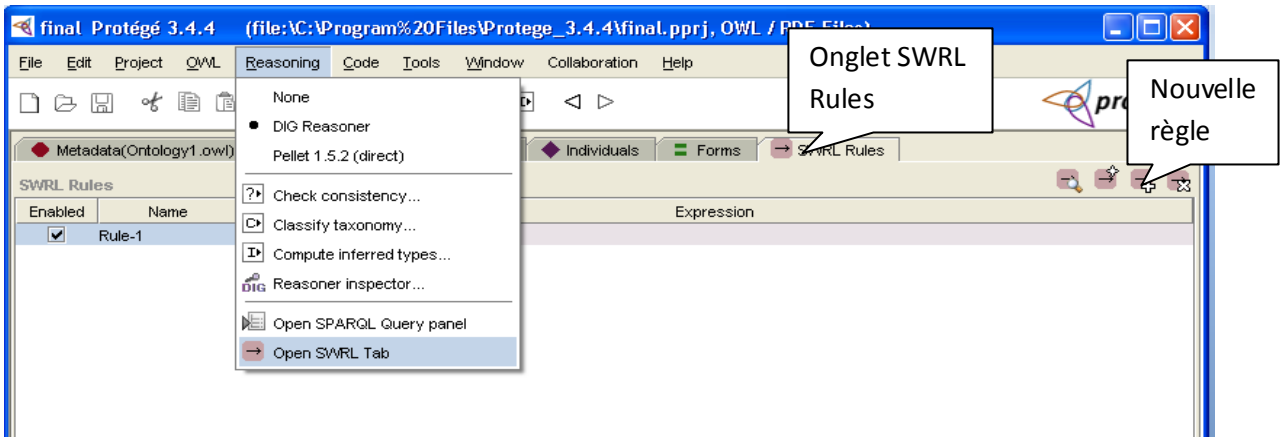


Figure10 : onglet SWRL

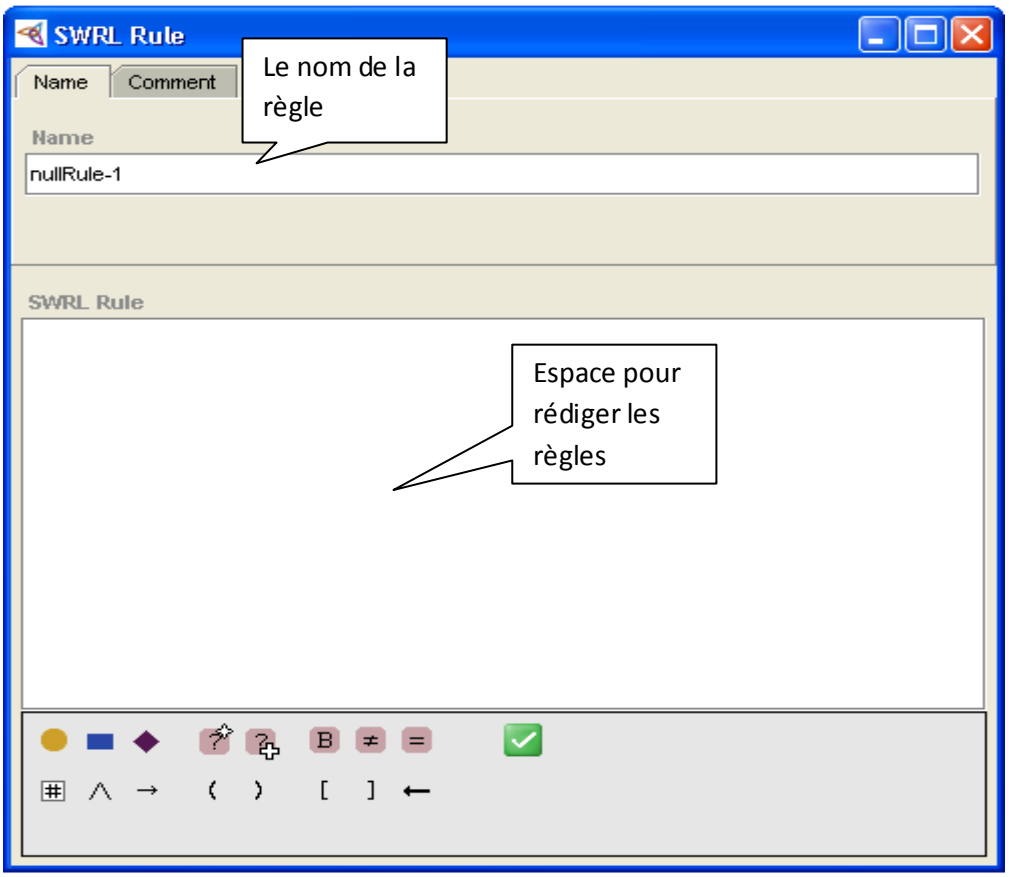


Figure11 : implémentation des règles pour les ontologies

X. Visualisation de notre ontologie et génération du code OWL :

Avec l'outil Protégé on peut visualiser graphiquement nos ontologies soit avec le plug-in Jambalaya qui est offert par Protégé ou bien par d'autre plug-in à télécharger sur internet comme GrapheViz qu'on peut télécharger via le site : http://www.01net.com/telecharger/windows/Internet/connection/fiches/30033.html

Et qu'on va intégrer ensuite dans Protégé. Mais avant d'introduire gapheViz il faut tout d'abord le sélectionner a partir de Projet/configuration/tab widgets.ensuite cocher les correspondante (OWLViz et Jambalaya).

V.1 Jumbalaya :

Jambalaya est une autre application de protégé qui permet la représentation de l'ontologie avec des graphes orientés. La figure suivante montre les sous classe de la classe erreur ainsi que leurs instances. Notons que, l'arc en bleu représente le lien « has subclass », celui en rouge représente le lien « has instance »

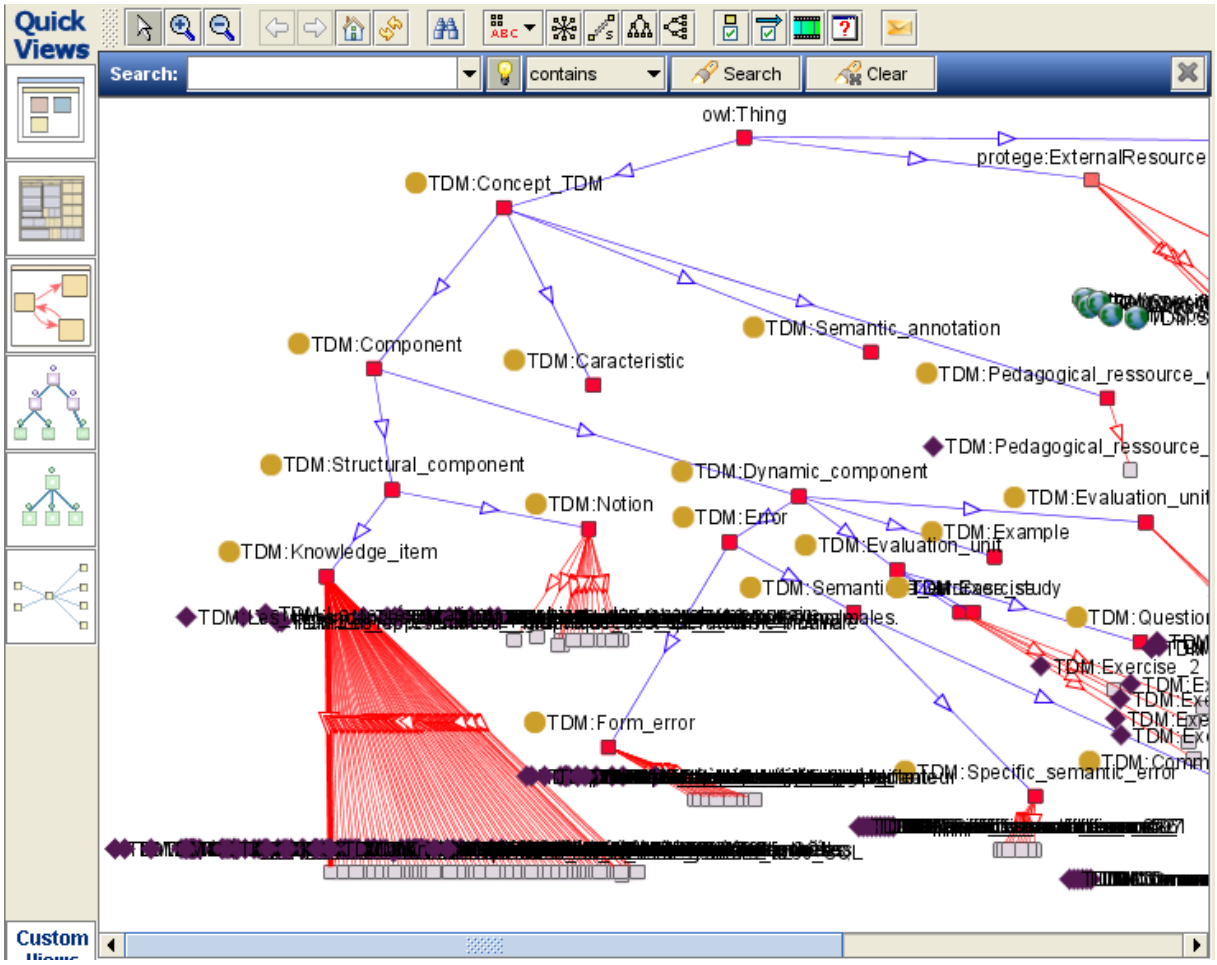


Figure12 : visualisation de l'ontologie avec Jambalaya.

V.2 OWLViz :

Pour pouvoir visualiser graphiquement les classes et la hiérarchie d'une ontologie OWL via grapheViz, il est nécessaire de lui greffer le DOT application qui permet la génération des graphes et que se trouve dans Graphviz/bin/dot.exe. On note que, la version de Graphviz doit être compatible avec celle du Protégé. Dans notre cas on a utilisé les

versions 2.8 et 3.4.4 respectivement. On montra c'est dessous comment faire intégrer Graphviz dans Protégé.

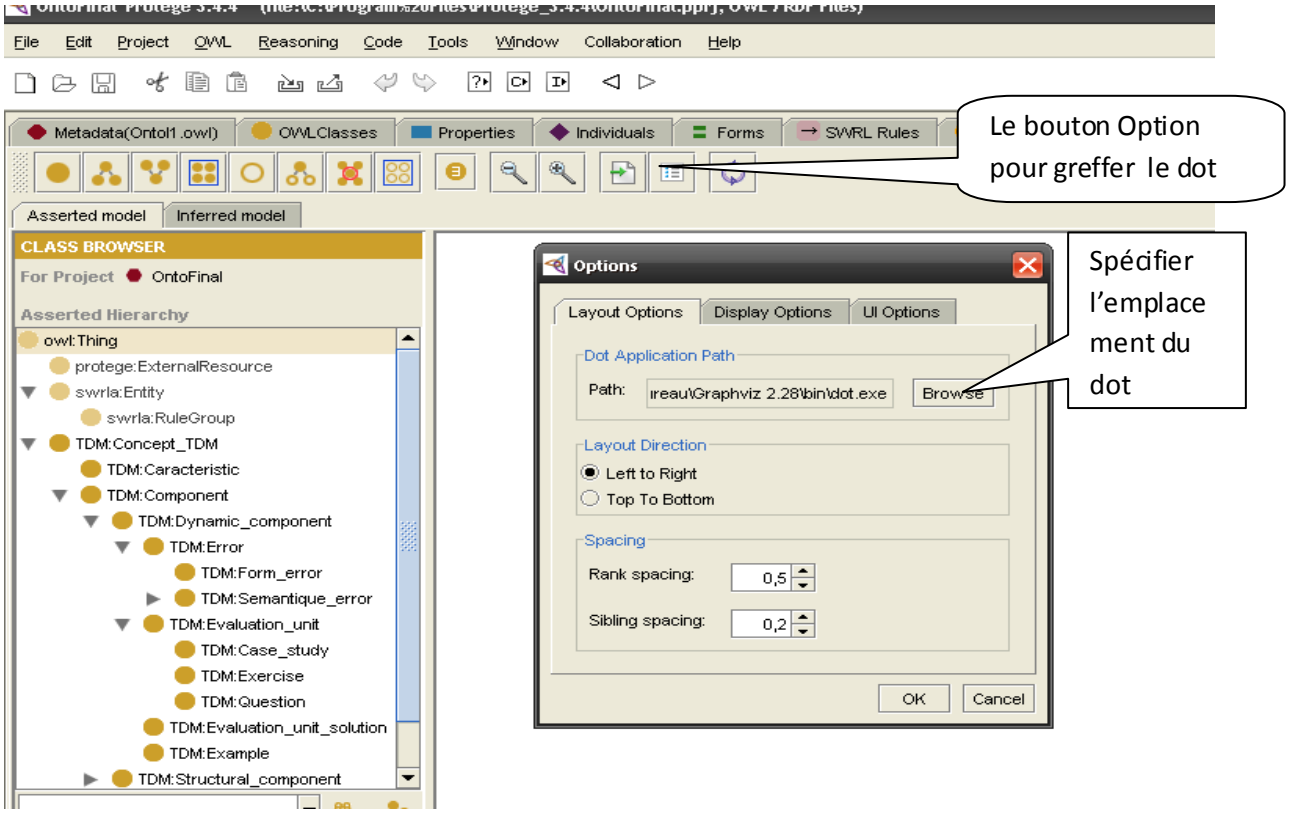


Figure13 : comment introduire Graphviz dans Protégé.

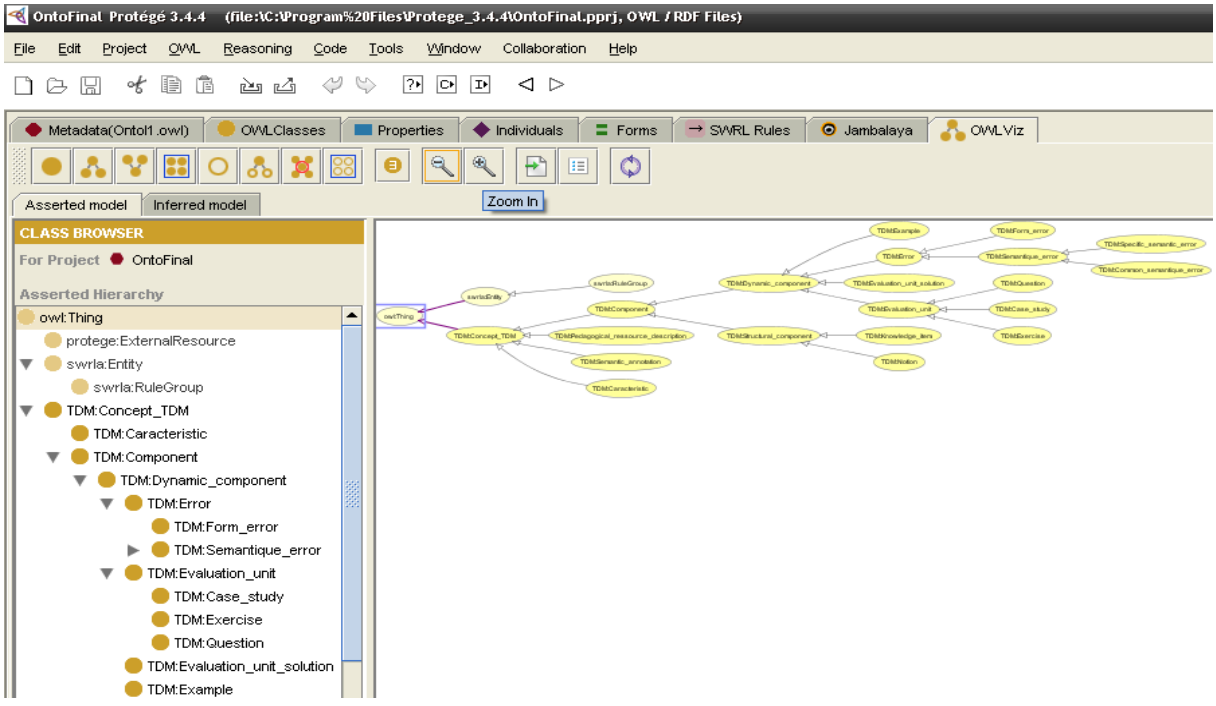


Figure14 : visualisation de l'ontologie avec OWLViz.

V.3 génération du code OWL :

Après avoir terminé l'édition de notre ontologie (les classes et les relations du diagramme UML, leurs individus et toutes les règles d'inférences nous avons généré le code OWL à partir de : File/Export format/OWL, dont voici un extrait de ce code :

Voici donc le code OWL de notre ontologie :

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
   xmlns:TDM="http://www.owl-ontologies.com/Ontol1.owl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#"
  xml:base="http://www.owl-ontologies.com/Ontol1.owl">
  <owl:Ontology rdf:about="">
   <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>
   <owl:imports rdf:resource="http://sqwrl.stanford.edu/ontologies/built-
ins/3.4/sqwrl.owl"/>
  </owl:Ontology>
  <owl:Class rdf:ID="Structural_component">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Component"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Notion">
   <rdfs:subClassOf rdf:resource="#Structural_component"/>
  </owl:Class>
  <owl:Class rdf:ID="Pedagogical_ressource_description">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Concept_TDM"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Semantique_error">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Error"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Evaluation_unit">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Dynamic_component"/>
   </rdfs:subClassOf>
```

Début de l'ontologie

L'entête

Sous classe de component

Définition de la classe structural component

```
</owl:Class>
<owl:Class rdf:ID="Knowledge_item">
 <rdfs:subClassOf rdf:resource="#Structural_component"/>
</owl:Class>
<owl:Class rdf:ID="Example">
 <rdfs:subClassOf>
  <owl:Class rdf:about="#Dynamic_component"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Caracteristic">
 <rdfs:subClassOf rdf:resource="#Concept_TDM"/>
</owl:Class>
<owl:Class rdf:ID="Common_semantique_error">
 <rdfs:subClassOf rdf:resource="#Semantique_error"/>
</owl:Class>
<owl:Class rdf:ID="Evaluation_unit_solution">
 <rdfs:subClassOf>
  <owl:Class rdf:about="#Dynamic_component"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Exercise">
 <rdfs:subClassOf rdf:resource="#Evaluation_unit"/>
</owl:Class>
<owl:Class rdf:about="#Error">
 <rdfs:subClassOf>
  <owl:Class rdf:about="#Dynamic_component"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Specific_semantic_error">
 <rdfs:subClassOf rdf:resource="#Semantique_error"/>
</owl:Class>
<owl:Class rdf:ID="Case_study">
 <rdfs:subClassOf rdf:resource="#Evaluation_unit"/>
</owl:Class>
<owl:Class rdf:ID="Form_error">
 <rdfs:subClassOf rdf:resource="#Error"/>
</owl:Class>
<owl:Class rdf:ID="Question">
 <rdfs:subClassOf rdf:resource="#Evaluation_unit"/>
</owl:Class>
<owl:Class rdf:ID="Semantic_annotation">
 <rdfs:subClassOf rdf:resource="#Concept_TDM"/>
</owl:Class>
<owl:Class rdf:about="#Component">
 <rdfs:subClassOf rdf:resource="#Concept_TDM"/>
</owl:Class>
<owl:Class rdf:about="#Dynamic_component">
 <rdfs:subClassOf rdf:resource="#Component"/>
</owl:Class>
```

```
<owl:ObjectProperty rdf:ID="Annotate_sem">
 <rdfs:domain rdf:resource="#Semantic_annotation"/>
 <rdfs:range>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Pedagogical_ressource_description"/>
    <owl:Class rdf:about="#Component"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Has_potential_error">
 <rdfs:domain rdf:resource="#Knowledge_item"/>
 <rdfs:range rdf:resource="#Error"/>
 <owl:inverseOf>
  <owl:ObjectProperty rdf:ID="Is_related_to"/>
 </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Evaluated_by">
 <rdfs:domain rdf:resource="#Knowledge_item"/>
 <owl:inverseOf>
  <owl:ObjectProperty rdf:ID="Defined_for"/>
 </owl:inverseOf>
 <rdfs:range rdf:resource="#Evaluation_unit"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="N_Composed_of_N">
 <rdfs:domain rdf:resource="#Notion"/>
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
 <rdfs:range rdf:resource="#Notion"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="N_Composed_of_KI">
 <rdfs:domain rdf:resource="#Notion"/>
 <rdfs:range rdf:resource="#Knowledge_item"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Is_solution_of">
 <rdfs:domain rdf:resource="#Evaluation_unit_solution"/>
 <owl:inverseOf>
  <owl:ObjectProperty rdf:ID="Has_solution"/>
 </owl:inverseOf>
 <rdfs:range rdf:resource="#Evaluation_unit"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Is">
 <rdfs:range rdf:resource="#Component"/>
 <rdfs:domain rdf:resource="#Pedagogical_ressource_description"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#Is_related_to">
 <owl:inverseOf rdf:resource="#Has_potential_error"/>
 <rdfs:domain rdf:resource="#Error"/>
 <rdfs:range rdf:resource="#Knowledge_item"/>
```

Déclaration de la relation Evaluated-by

Son domaine

La relation inverse

Son rang

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Has_example">
 <rdfs:domain rdf:resource="#Structural_component"/>
 <owl:inverseOf>
  <owl:ObjectProperty rdf:ID="Is_example_of"/>
 </owl:inverseOf>
 <rdfs:range rdf:resource="#Example"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Annotates">
 <rdfs:domain rdf:resource="#Component"/>
 <rdfs:range rdf:resource="#Pedagogical_ressource_description"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Required_for">
 <rdfs:domain rdf:resource="#Structural_component"/>
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
 <rdfs:range rdf:resource="#Structural_component"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#Has_solution">
 <rdfs:range rdf:resource="#Evaluation_unit_solution"/>
 <owl:inverseOf rdf:resource="#Is_solution_of"/>
 <rdfs:domain rdf:resource="#Evaluation_unit"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#Defined_for">
 <rdfs:range rdf:resource="#Knowledge_item"/>
 <owl:inverseOf rdf:resource="#Evaluated_by"/>
 <rdfs:domain rdf:resource="#Evaluation_unit"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Caracterise">
 <rdfs:domain rdf:resource="#Caracteristic"/>
 <rdfs:range>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Evaluation_unit"/>
    <owl:Class rdf:about="#Knowledge_item"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#Is_example_of">
 <rdfs:domain rdf:resource="#Example"/>
 <rdfs:range rdf:resource="#Structural_component"/>
 <owl:inverseOf rdf:resource="#Has_example"/>
</owl:ObjectProperty>
```

```
<swrl:Imp rdf:about="Rule-2">
 <swrl:head>
  <swrl:AtomList>
   <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
   <rdf:first>
    <swrl:IndividualPropertyAtom>
     <swrl:argument1>
      <swrl:Variable rdf:about="x"/>
     </swrl:argument1>
     <swrl:argument2>
      <swrl:Variable rdf:about="z"/>
     </swrl:argument2>
     <swrl:propertyPredicate rdf:resource="#Required_for"/>
    </swrl:IndividualPropertyAtom>
   </rdf:first>
  </swrl:AtomList>
</swrl:head>
 <swrl:body>
  <swrl:AtomList>
   <rdf:first>
    <swrl:IndividualPropertyAtom>
     <swrl:argument1 rdf:resource="x"/>
     <swrl:propertyPredicate rdf:resource="#Required_for"/>
     <swrl:argument2>
      <swrl:Variable rdf:about="y"/>
     </swrl:argument2>
    </swrl:IndividualPropertyAtom>
   </rdf:first>
   <rdf:rest>
    <swrl:AtomList>
     <rdf:first>
      <swrl:IndividualPropertyAtom>
       <swrl:argument2 rdf:resource="z"/>
       <swrl:argument1 rdf:resource="y"/>
       <swrl:propertyPredicate rdf:resource="#Required_for"/>
      </swrl:IndividualPropertyAtom>
     </rdf:first>
     <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </swrl:AtomList>
   </rdf:rest>
  </swrl:AtomList>
 </swrl:body>
</swrl:Imp>
<swrl:Imp rdf:about="Rule-1">
 <swrl:head>
  <swrl:AtomList>
   <rdf:first>
    <swrl:IndividualPropertyAtom>
     <swrl:argument1 rdf:resource="x"/>
```

L'entête

Corp.

Définition de la Règle N° 02

```
        <swrl:argument2 rdf:resource="z"/>
        <swrl:propertyPredicate rdf:resource="#N_Composed_of_N"/>
       </swrl:IndividualPropertyAtom>
     </rdf:first>
     <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </swrl:AtomList>
   </swrl:head>
   <swrl:body>
    <swrl:AtomList>
     <rdf:first>
      <swrl:IndividualPropertyAtom>
       <swrl:argument2 rdf:resource="y"/>
       <swrl:argument1 rdf:resource="x"/>
       <swrl:propertyPredicate rdf:resource="#N_Composed_of_N"/>
      </swrl:IndividualPropertyAtom>
     </rdf:first>
     <rdf:rest>
      <swrl:AtomList>
       <rdf:first>
        <swrl:IndividualPropertyAtom>
         <swrl:propertyPredicate rdf:resource="#N_Composed_of_N"/>
         <swrl:argument1 rdf:resource="y"/>
         <swrl:argument2 rdf:resource="z"/>
        </swrl:IndividualPropertyAtom>
       </rdf:first>
       <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      </swrl:AtomList>
     </rdf:rest>
    </swrl:AtomList>
   </swrl:body>
  </swrl:Imp>
  <swrl:Imp rdf:about="Rule-3">
   <swrl:head>
    <swrl:AtomList>
     <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
     <rdf:first>
      <swrl:ClassAtom>
       <swrl:argument1 rdf:resource="x"/>
       <swrl:classPredicate rdf:resource="#Component"/>
      </swrl:ClassAtom>
     </rdf:first>
    </swrl:AtomList>
   </swrl:head>
   <swrl:body>
    <swrl:AtomList>
     <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
     <rdf:first>
      <swrl:ClassAtom>
       <swrl:argument1 rdf:resource="x"/>
```

```
        <swrl:classPredicate rdf:resource="#Structural_component"/>
       </swrl:ClassAtom>
      </rdf:first>
    </swrl:AtomList>
  </swrl:body>
</swrl:Imp>
<rdf:Description rdf:about="http://www.owl-ontologies.com/Rule-1">
  <swrl:head>
    <swrl:AtomList>
      <rdf:first>
       <swrl:ClassAtom>
        <swrl:argument1>
         <swrl:Variable rdf:about="http://www.owl-ontologies.com/x"/>
        </swrl:argument1>
        <swrl:classPredicate rdf:resource="#Component"/>
       </swrl:ClassAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </swrl:AtomList>
  </swrl:head>
  <swrl:body>
    <swrl:AtomList>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      <rdf:first>
       <swrl:ClassAtom>
        <swrl:argument1 rdf:resource="http://www.owl-ontologies.com/x"/>
        <swrl:classPredicate rdf:resource="#Structural_component"/>
       </swrl:ClassAtom>
      </rdf:first>
    </swrl:AtomList>
  </swrl:body>
</rdf:Description>
<swrl:Imp rdf:about="Rule-8">
  <swrl:head>
    <swrl:AtomList>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      <rdf:first>
       <swrl:IndividualPropertyAtom>
        <swrl:argument1 rdf:resource="z"/>
        <swrl:argument2 rdf:resource="x"/>
        <swrl:propertyPredicate rdf:resource="#Required_for"/>
       </swrl:IndividualPropertyAtom>
      </rdf:first>
    </swrl:AtomList>
  </swrl:head>
  <swrl:body>
    <swrl:AtomList>
      <rdf:first>
       <swrl:IndividualPropertyAtom>
```

```
        <swrl:argument1 rdf:resource="x"/>
        <swrl:argument2 rdf:resource="y"/>
        <swrl:propertyPredicate rdf:resource="#N_Composed_of_KI"/>
       </swrl:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest>
       <swrl:AtomList>
        <rdf:first>
         <swrl:IndividualPropertyAtom>
          <swrl:propertyPredicate rdf:resource="#Required_for"/>
          <swrl:argument2 rdf:resource="y"/>
          <swrl:argument1 rdf:resource="z"/>
         </swrl:IndividualPropertyAtom>
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
       </swrl:AtomList>
      </rdf:rest>
     </swrl:AtomList>
    </swrl:body>
  </swrl:Imp>
  <swrl:Imp rdf:about="lRule-4">
    <swrl:body>
     <swrl:AtomList>
      <rdf:rest>
       <swrl:AtomList>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        <rdf:first>
         <swrl:IndividualPropertyAtom>
          <swrl:argument1 rdf:resource="y"/>
          <swrl:argument2 rdf:resource="z"/>
          <swrl:propertyPredicate rdf:resource="#N_Composed_of_KI"/>
         </swrl:IndividualPropertyAtom>
        </rdf:first>
       </swrl:AtomList>
      </rdf:rest>
      <rdf:first>
       <swrl:IndividualPropertyAtom>
        <swrl:propertyPredicate rdf:resource="#N_Composed_of_N"/>
        <swrl:argument2 rdf:resource="y"/>
        <swrl:argument1 rdf:resource="x"/>
       </swrl:IndividualPropertyAtom>
      </rdf:first>
     </swrl:AtomList>
    </swrl:body>
    <swrl:head>
     <swrl:AtomList>
      <rdf:first>
       <swrl:IndividualPropertyAtom>
        <swrl:propertyPredicate rdf:resource="#N_Composed_of_KI"/>
```

```
        <swrl:argument2 rdf:resource="z"/>
        <swrl:argument1 rdf:resource="x"/>
       </swrl:IndividualPropertyAtom>
     </rdf:first>
     <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
   </swrl:AtomList>
  </swrl:head>
</swrl:Imp>
<swrl:Imp rdf:about="Rule-6">
  <swrl:body>
   <swrl:AtomList>
    <rdf:rest>
     <swrl:AtomList>
      <rdf:first>
       <swrl:IndividualPropertyAtom>
        <swrl:argument1 rdf:resource="y"/>
        <swrl:argument2 rdf:resource="z"/>
        <swrl:propertyPredicate rdf:resource="#Has_potential_error"/>
       </swrl:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
     </swrl:AtomList>
    </rdf:rest>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="#N_Composed_of_KI"/>
      <swrl:argument1 rdf:resource="x"/>
      <swrl:argument2 rdf:resource="y"/>
     </swrl:IndividualPropertyAtom>
    </rdf:first>
   </swrl:AtomList>
  </swrl:body>
  <swrl:head>
   <swrl:AtomList>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:argument2 rdf:resource="z"/>
      <swrl:argument1 rdf:resource="x"/>
      <swrl:propertyPredicate rdf:resource="#Has_potential_error"/>
     </swrl:IndividualPropertyAtom>
    </rdf:first>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
   </swrl:AtomList>
  </swrl:head>
</swrl:Imp>
<swrl:Imp rdf:about="Rule-7">
  <swrl:body>
   <swrl:AtomList>
    <rdf:first>
```

```
      <swrl:IndividualPropertyAtom>
       <swrl:argument1 rdf:resource="x"/>
       <swrl:argument2 rdf:resource="y"/>
       <swrl:propertyPredicate rdf:resource="#N_Composed_of_N"/>
      </swrl:IndividualPropertyAtom>
     </rdf:first>
     <rdf:rest>
      <swrl:AtomList>
       <rdf:first>
        <swrl:IndividualPropertyAtom>
         <swrl:argument1 rdf:resource="z"/>
         <swrl:argument2 rdf:resource="y"/>
         <swrl:propertyPredicate rdf:resource="#Required_for"/>
        </swrl:IndividualPropertyAtom>
       </rdf:first>
       <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      </swrl:AtomList>
     </rdf:rest>
    </swrl:AtomList>
   </swrl:body>
   <swrl:head>
    <swrl:AtomList>
     <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
     <rdf:first>
      <swrl:IndividualPropertyAtom>
       <swrl:argument1 rdf:resource="z"/>
       <swrl:argument2 rdf:resource="x"/>
       <swrl:propertyPredicate rdf:resource="#Required_for"/>
      </swrl:IndividualPropertyAtom>
     </rdf:first>
    </swrl:AtomList>
   </swrl:head>
  </swrl:Imp>
  <swrl:Imp rdf:about="Rule-5">
   <swrl:head>
    <swrl:AtomList>
     <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
     <rdf:first>
      <swrl:IndividualPropertyAtom>
       <swrl:argument1 rdf:resource="x"/>
       <swrl:argument2 rdf:resource="z"/>
       <swrl:propertyPredicate rdf:resource="#Evaluated_by"/>
      </swrl:IndividualPropertyAtom>
     </rdf:first>
    </swrl:AtomList>
   </swrl:head>
   <swrl:body>
    <swrl:AtomList>
     <rdf:rest>
```

```
      <swrl:AtomList>
       <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
       <rdf:first>
        <swrl:IndividualPropertyAtom>
         <swrl:argument1 rdf:resource="y"/>
         <swrl:argument2 rdf:resource="z"/>
         <swrl:propertyPredicate rdf:resource="#Evaluated_by"/>
        </swrl:IndividualPropertyAtom>
       </rdf:first>
      </swrl:AtomList>
     </rdf:rest>
     <rdf:first>
      <swrl:IndividualPropertyAtom>
       <swrl:argument2 rdf:resource="y"/>
       <swrl:argument1 rdf:resource="x"/>
       <swrl:propertyPredicate rdf:resource="#N_Composed_of_KI"/>
      </swrl:IndividualPropertyAtom>
     </rdf:first>
    </swrl:AtomList>
   </swrl:body>
  </swrl:Imp>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.4.4, Build 579) http://protege.stanford.edu -->
```

Figure15 : notre ontologie sous fichier OWL.

Conclusion :

Ce chapitre a été consacré à la représentation de notre ontologie avec le langage UML, ainsi son correspondance en OWL, puis son édition et sa visualisation sous l'outil Protégé.

Dans le prochain chapitre nous allons procède à l'exploitation de notre ontologie sous l'outil Protégé et sous un environnement Java.

CHAPITRE III

Exploitation de l'ontologie.

Introduction :

Dans ce troisième chapitre nous allons entamer la troisième étape de notre travail qui consiste en « exploitation de notre ontologie de domaine d'enseignement des bases de données relationnel »

Pour cela nous avons utilisé des outils en l'occurrence Protégé et le langage SPARQL pour exploité notre ontologie sous protégé, et NetBeans et la bibliothèque Jena comme environnement pour son exploitation sous un environnement java.

Dans un premier lieu nous allons présenter ces différents outils, en suite nous allons présenter l'exploitation de l'ontologie soit sous protégé ou sous java.

IV. Outils et langages utilisés :

Afin d'exploiter notre ontologie nous avons utilisé plusieurs outils et langages soit dans l'exploitation sous Protégé ou sous Java que nous citons :

I.1 les outils :

I.1.1 Protégé :

Protégé **[STA, 05]** est un éditeur d'ontologies distribué en open source par l'université en informatique médicale de Stanford. Protégé n'est pas un outil spécialement dédié à OWL, mais un éditeur hautement extensible, capable de manipuler des formats très divers. Le support d'OWL, comme de nombreux autres formats, est possible dans Protégé grâce à un plugin dédié.

Protégé permet de guider les développeurs et les experts dans le processus de développement de système à base de connaissances. C'est un logiciel libre d'utilisation qui peut être modifié par l'utilisateur. Il est alors possible de réaliser des modules additionnels pour modifier ou compléter ce logiciel. Nous avons choisi la version 3.4.4 pour éditer notre ontologie.

I.1.2 GraphViz :[2]

L'application GraphViz permet de représenter graphiquement des graphes. Elle a été conçue par une équipe des laboratoires de recherche de AT&T (American Telephone & Telegraph).

Cette application convient à la représentation de graphes très denses comprenant un très grand nombre de nœuds grâce des algorithmes très puissants. Elle est très rapide à l'exécution et le rendu est optimisé afin que les liens ne recouvrent pas les nœuds et qu'ils ne se croisent pas.

De plus, entièrement paramétrable, l'application permet de personnaliser le rendu des graphes par le choix des formes, couleurs et polices de caractères. Le format des fichiers d'entrée est simple et souple. Les formats de sortie sont très variés. Et la version que nous avons choisi est la version 2.8 qui compatible avec l'éditeur Protégé 3.4.4.

I.1.3 Netbeans : [3]

NetBeans est un environnement de développement intégré (EDI), placé en *open source* par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License). En

plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web). Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris, Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Développent Kit JDK est requis pour les développements en Java. L'IDE Netbeans s'enrichit à l'aide de plugins. Netbeans été le logiciel de développement de notre application Web (servelet et JSP).

I.1.4 API Jena : [4]

JENA est un outil open source développé par une équipe de la firme HP (Hewlett Packard) dans le cadre du Projet HP « Labs Semantic Web Programme» qui a pour but de réaliser un outil d'exploitation des fichiers OWL. Jena est une API java, comprend des raisonneurs intégrés mais permet également d'utiliser des raisonneurs externes. Jena comportes les outils suivantes :

- Une API pour le langage RDF
- Un module de lecture/ écriture sur les serveurs RDF : RDF/XML, N3et NTRIPLE
- Une API pour le langage OWL
- RDQL : un langage d'interrogation des ontologies Implémenter en RDF et OWL.

Dans notre cas on a utilisé Jena version 2.3 qui a était la version la plus compatible avec l'ensemble des logiciels utilisés pour la réalisation de notre projet

I.1.5 Dreamwaver : [5]

Adobe Dreamweaver (anciennement **Macromedia Dreamweaver**) est un éditeur de site web. Il fut l'un des premiers éditeurs HTML de type « tel affichage, tel résultat », mais également l'un des premiers à intégrer un gestionnaire de site (CyberStudio GoLive étant le premier). Ces innovations l'imposèrent rapidement comme l'un des principaux éditeurs de site web, aussi bien utilisable par le néophyte que par le professionnel. Dreamweaver offre deux modes de conception par son menu affichage. L'utilisateur peut choisir entre un mode création permettant d'effectuer la mise en page directement à

l'aide d'outils simples, comparables à un logiciel de traitement de texte (insertion de tableau, d'image, etc.). Il est également possible d'afficher et de modifier directement le code (HTML ou autre) qui compose la page. On peut passer très facilement d'un mode d'affichage à l'autre, ou opter pour un affichage mixte. Cette dernière option est particulièrement intéressante pour les débutants qui, à terme, souhaitent se familiariser avec le langage HTML. Dreamweaver a évolué avec les technologies de l'internet. Il offre aujourd'hui la possibilité de concevoir des feuilles de style. Les liaisons avec des bases de données ont également été améliorées ainsi que le chargement des fichiers sur les serveurs d'hébergement. Il propose en outre l'utilisation de modèles imbriqués de pages web, selon un format propriétaire.

Dans notre projet nous avons utilisé Dreamweaver pour réaliser les pages JSP que nous allons importer ensuite vers netbeans.

I.2 les langages :

I.2.1 Langage SPARQL : [MBB, 2009]

SPARQL (Protocol And Rdf Query Language) du W3C et plus spécifiquement défini par le groupe de travail RDF Data Access, est à la fois un langage et un protocole. Le protocole va notamment permettre à un client Web de consulter au travers d'une requête SPARQL, un service ou point d'accèsSPARQL (endpoint aussi nommé processor en anglais). Le service en question, va traiter la requête et retourner le résultat dans déférents formats (HTML, XML, RDF/XML, N3, . . .). Le protocole SPARQL s'appuie sur WSDL (Web Service Description Language) qui normalise la description des services Web et de leurs accès.

Forme des requêtes SPARQL (les clauses rappellent pour certaines, SQL et les langages déclaratifs) :

PREFIX indique l'adresse (espace de noms) d'un schéma pouvant être exploité ensuite dans la construction de la requête

Syntaxe d'une requête SPARQL

PREFIX nom_prefix:<prefix>

SELECT ?subject ?object

WHERE {?subject nom_prefix: predicat ?object }

SELECT... **[FROM]...WHERE** retourne les ressources qui sont associées aux variables liées dans la clause **WHERE**

CONSTRUCT engendre un nouveau graphe qui complète le graphe interrogé (ancêtre, tante, oncle etc. dans un graphe contenant des informations généalogiques). **CONSTRUCT** va donc permettre d'exploiter des requêtes dites constructives (**SELECT** exploite, à l'inverse, des requêtes dites interrogatives)

UNION graphes alternatifs (correspond à au moins un des graphes précises)

FILTER rajouter des conditions devant être satisfaites.

DESCRIBE retourne une description des ressources satisfaisant la requête.

OPTIONAL pattern(s) de graphe optionnel(s).

ASK évalue si la requête va retourner un ensemble de ressources ou bien l'ensemble vide.

Exemple :

- SELECT ?individu
  WHERE { ?individu rdf :type TDM : Knowledge_item.}

I.2.2 Les Servlets (Langage Java) : [POU, 2009]

Une servlet est un programme java qui utilise des modules supplémentaires figurant dans l'API jena. Elle s'exécute dynamiquement sur le serveur Web et permet l'extension des fonctions de serveur. Les servlets permettent donc de gérer des requêtes HTTP et de fournir au client une réponse HTTP dynamique (donc de créer des pages Web dynamiques).

Une servlet s'exécute dans un conteneur de servlet utilisé pour établir le lien entre la servlet et le serveur Web, dans notre cas, on utilisé le conteneur Glassfish version 3.0 qui est intégrer dans Netbeans Ainsi le programmeur n'a pas à se soucier des

détails techniques tels que la connexion au réseau, la mise en forme de la réponse à la norme http.

Voici un exemple de servlet classique :

```
import javax.servlet.* ;
import java.io.* ;

public class HelloServlet extends GenericServlet
{
  public void service (HttpServletRequest request, HttpServletResponse
response)
  {
    try
    {
      PrintWriter out = response.getWriter() ;
      out.println ("<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML
4.01//EN\">") ;
      out.println ("<title>Bonjour tout le monde !</title>") ;
      out.println ("<p>Hello world!</p>") ;
    }
    catch (IOException e)
    {
      e.printStackTrace() ;
    }
  }
}
```

Figure1 : Exemple de servlet.

I.2.3 langage HTML : [6]

L'***Hypertext Markup Language***, généralement abrégé **HTML**, est le format de données conçu pour représenter les pages web. C'est un langage de balisage qui permet d'écrire de l'hypertexte, d'où son nom. HTML permet également de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des éléments programmables tels que des *applets*. Il permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web. Il est souvent utilisé conjointement avec des langages de programmation (JavaScript) et des formats de présentation (feuilles de style en cascade). HTML est initialement dérivé du *Standard Generalized Markup Language* (SGML). Du HTML a été utilisé dans Dreamweaver pour la création des pages web que nous appelons des JSP lorsque nous les importerons dans Netbeans

V. Exploitation de l'ontologie sous l'outil Protégé :

Protégé offre la possibilité d'exploiter les base de connaissances des ontologies à l'aide d'un panneau d'exécution de requête appelé SPARQL Query panel qu'on peut obtenir à partir de Raisonning/Open SPARQL Query Panel comme le montre la figure suivante :

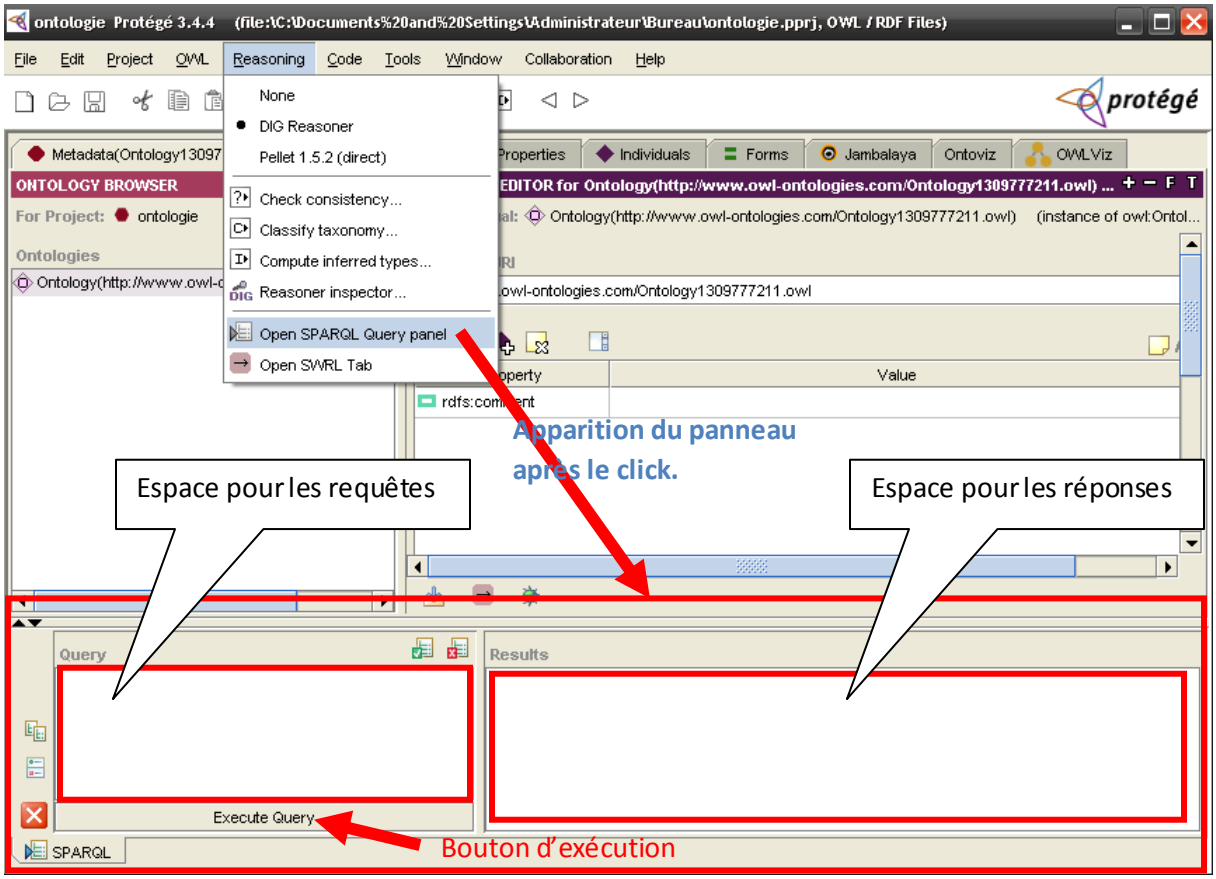


Figure16 : ouverture de panneau d'exécution de requêtes sous Protégé.

II.1 L'interprétation des requêtes sous SPARQL et leur exécution sous Protégé:

➢ **Afficher tous les individus de la classe Knowledge_item.**

- SELECT ?individu
  WHERE { ?individu rdf :type TDM : Knowledge_item.}

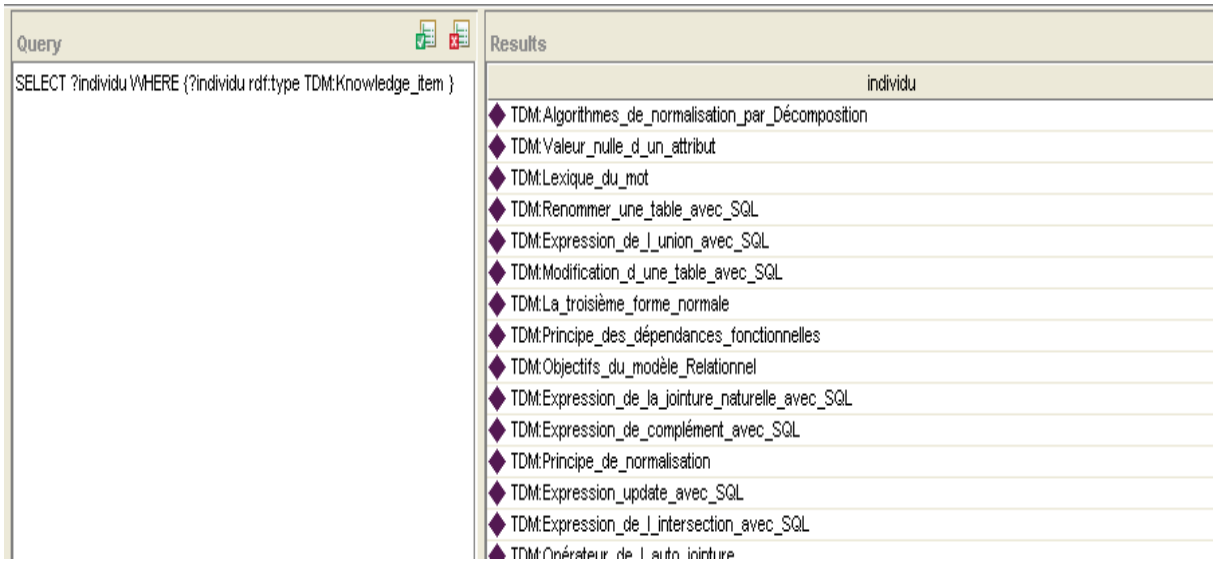


Figure17 : Exécution de la requête 1 sous protégé.

- **Afficher tous les individus de la classe Notion.**

  SELECT ?individu

  WHERE {?individu rdf:type TDM:Notion }

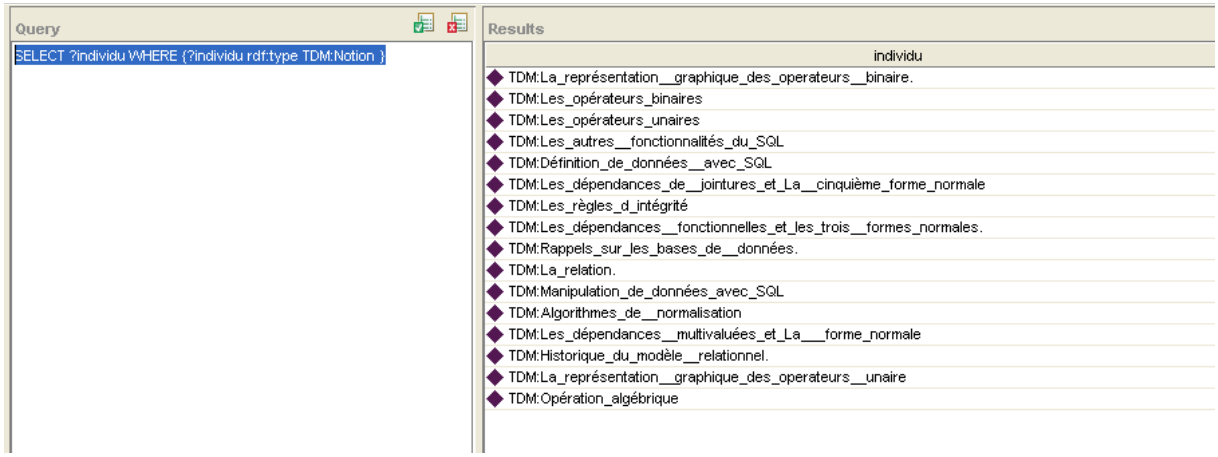


Figure18 : Exécution de la requête 2 sous protégé.

- **Sélectionner les erreurs reliées à chaque item de connaissance.**

- SELECT ?Error ?Knowledge_item

  WHERE { ? Knowledge_item TDM : has_potential_error ?Error.}

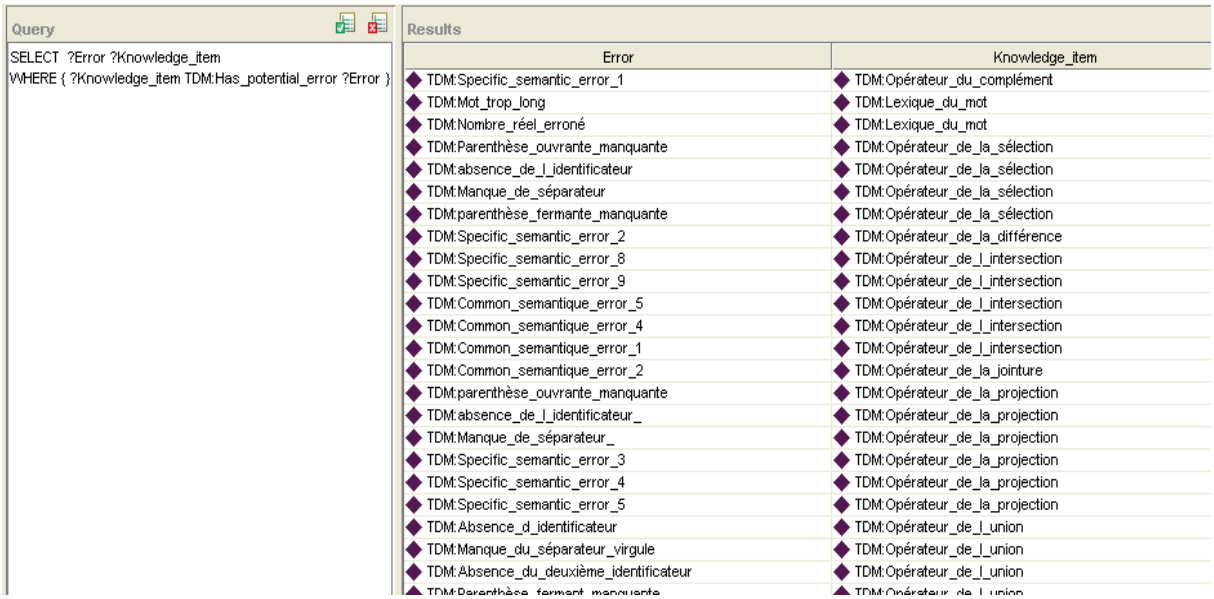


Figure19 : Exécution de la requête 3 sous protégé.

- **Sélectionner tous les exercices, leurs solutions, ainsi les items de connaissance pour lesquels ils sont défini.**

- SELECT ?Exercise ?Evaluation_unit_solution ?Knowledge_item
  WHERE { ?Exercise TDM : has_solution ?Evaluation_unit_solution.
  ? Exercise TDM : defined_for ?Knowledge_item. }

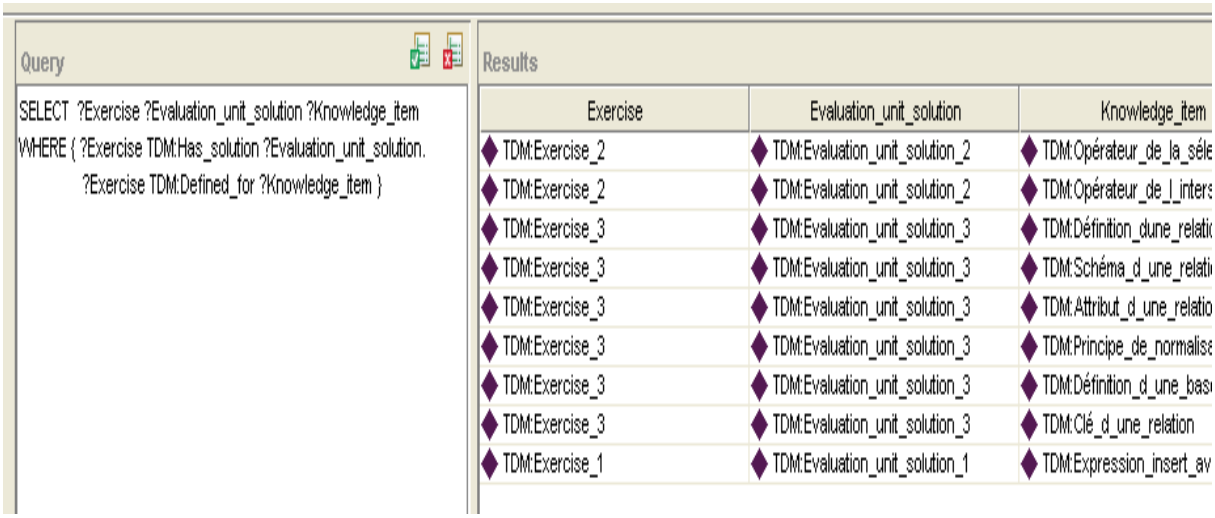


Figure20 : Exécution de la requête 4 sous protégé.

- **Toutes les erreurs qui sont en relation avec l'item de connaissance "Operateur de l'union"**

- SELECT ?Error ?KI

WHERE {?KI TDM:Name_KI "Opérateur_de_l_union".

?KI TDM:Has_potential_error ?Error }

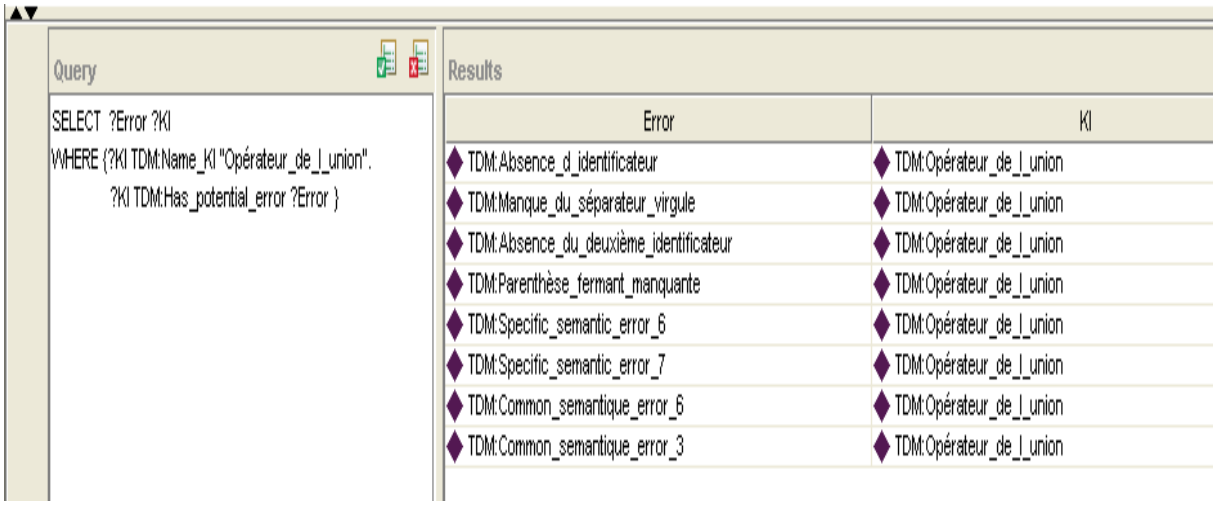


Figure21 : Exécution de la requête 5 sous protégé.

- **Tous les exercices ou l'erreur " Manque de séparateur " est potentielles.**

SELECT ?Exercise

WHERE {?Exercise TDM:Defined_for ?Knowledge_item.

?Knowledge_item TDM:Has_potential_error ?Form_error.

?Form_error TDM:Name_error_form "Manque de séparateur"

}

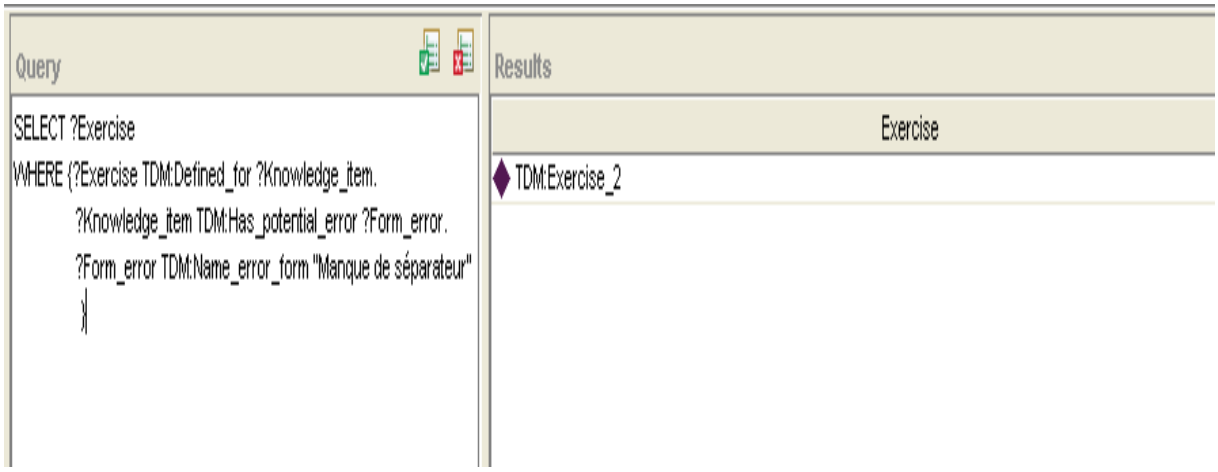


Figure22 : Exécution de la requête 6 sous protégé.

- **Tous les exercices qui évaluent l'item "** Expression_insert_avec_SQL**".**

- SELECT ?Exercise

WHERE {?Exercise TDM:Defined_for ?Knowledge_item.

?Knowledge_item TDM:Name_KI "Expression_insert_avec_SQL"

}

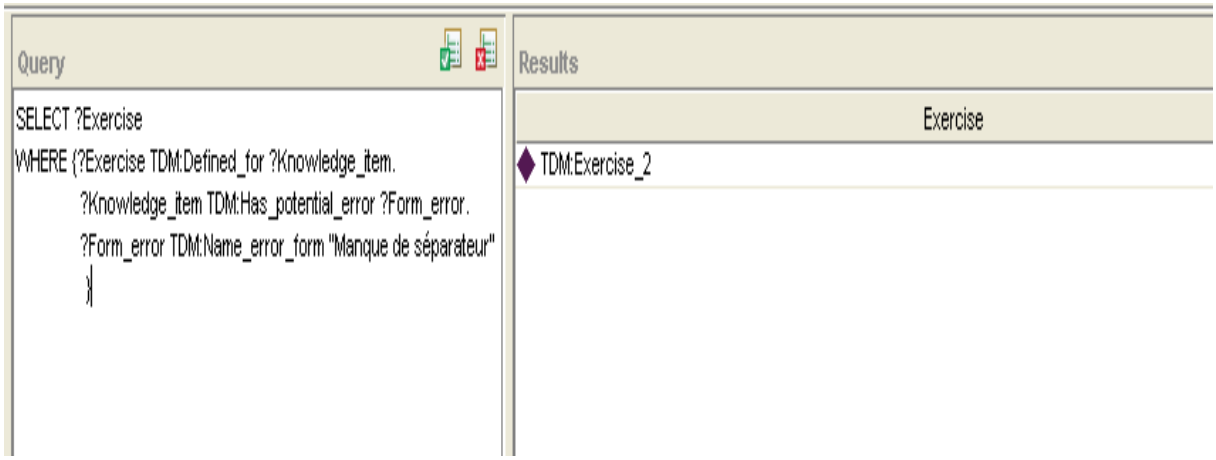


Figure23 : Exécution de la requête 7 sous protégé.

- **Les notions évaluées par un exo "Exercise1".**

- SELECT ?Notion

  WHERE {?Notion TDM:N_Composed_of_KI ?Knowledge_item.

  ?Exercise TDM:Name_exercise "Exercise_1".

  ?Exercise TDM:Defined_for ?Knowledge_item.

  }

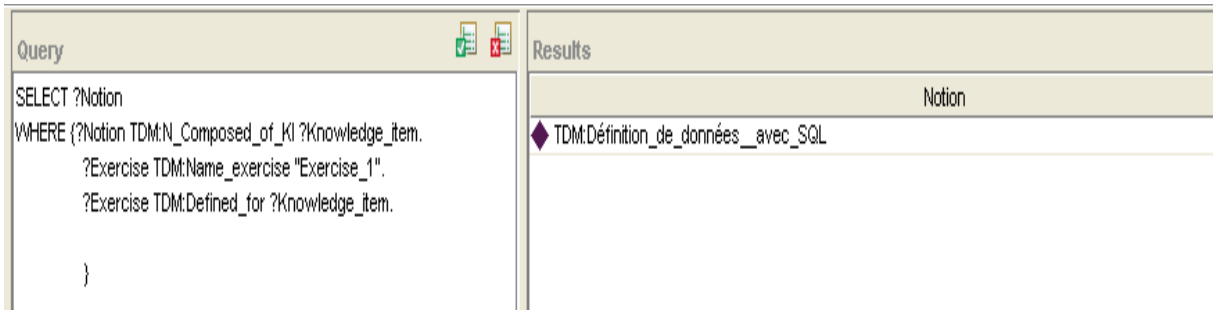


Figure24 : Exécution de la requête 8 sous protégé.

VI. Exploitations de l'ontologie dans un environnement Java :

III.1 Préparation de l'environnement de travail :

Avoir un environnement de travail complet pour le développement de toute application est essentiel. De fait la préparation de l'environnement de travail est une étape primordiale qu'il faut réaliser avec soin, pour éviter des problèmes lors du développement.

En effet notre environnement est préparé comme suit :

- **JDK** (Java Devellopment Kit) : téléchargeable sur le site de SUN Microsystems (fondateur du langage) **http://java.sun.com/javase/downloads/index.jsp. ensuite l'installé sur la machine.**
- **NetBeans :** nous avons utilisé la version 6.9 qui est gratuite sur site de SUN microsystems http://java.sun.com.
- **Logiciel Protégé :** nous avons utilisé la version 3.4.4 et les raisons qui nous ont poussés à opter pour Protégé sont :
  · C'est l'un des meilleurs éditeurs d'ontologie sur recommandation des experts du l'ingénierie ontologique.
  · L'éditeur de classe Protégé permet la création et la gestion des relations en termes de propriétés et de hiérarchies.
  · De même, il permet la visualisation graphique des classes et des hiérarchies OWL (Ontology Web Language) à l'aide du logiciel GraphViz
  . C'est un logiciel gratuit à plusieurs langues et il permet la génération de fichier à plusieurs formats.
- **La bibliothèque Jena :** qui va servir de passerelle entre l'application java et la base de connaissance tant que fichier OWL.
- **Serveur d'application :** glass fiche qui est inclus dans NetBeans

Ces outils nous les avons utilisés sous le système d'exploitation Windows XP, et Java EE comme langage de programmation pour les servlets et le langage SPARQL pour interroger la base de connaissance.

III.2 Example d'une servelet Java :

Lors de l'exécution d'une servlet celle-ci se connecte au fichier OWL qu'elle exécute avec l' éventuelle requête SPARQL incorporée et elle récupère les données.

Le code de la servlet suivante permet d'afficher tout les individus d'une classe et la partie sélectionnée montre le code de la requête.

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QueryFactory;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.vocabulary.RDF;
public class B extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse
response)
throws IOException, ServletException{
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>page1</title>");
out.println("</head>");
out.println("<body background=\"image/Bg-accuil-arriere plan.jpg\"
text=\"#000FFF\" link=\"#0000FF\"vlink=\"#0000FF\" alink=\"#FF0000\">");
out.println("<img src=\"image/bienvenu_entre.jpg\" width=\"1235\"
height=\"180\" border=\"0\"alt=\"\">");
//String classes= request.getParameter("classe");
out.println( "<center>");
String tod_file = "ontologie.owl";
String NL = System.getProperty("line.separator") ;
Model model = ModelFactory.createDefaultModel() ;
model.read("file:/E:/ontologie.owl");
//log.debug( "We have loaded a model with no. statements = " + m.size() );
```

```
String tod ="http://www.owl-ontologies.com/Ontology1309777211.owl#";
String prolog1 = "PREFIX tod: <"+tod+">" ;
String prolog2 = "PREFIX rdf: <"+RDF.getURI()+">" ;
// Query string.
String queryString = prolog1 + NL + prolog2 + NL +
"SELECT ?individu WHERE {?individu rdf:type tod:Knowledge_item }" ;
Query query = QueryFactory.create(queryString) ;
// Print with line numbers
query.serialize() ;
System.out.println() ;
// Create a single execution of this query, apply to a model
// which is wrapped up as a Dataset
QueryExecution qexec = QueryExecutionFactory.create(query, model) ;
// Or QueryExecutionFactory.create(queryString, model) ;
out.println("<marquee behavior=\"alternate\"><h1><center>La liste des
items de connaissances:Knowledge_item : </center></h1></marquee>") ;
out.println(" <div align=\"center\"><table border=\"0\"
cellpadding=\"10\"cellspacing=\"80\">");
out.println("<tr> <td><b> ");
try {
// Assumption: itâ€™s a SELECT query.
ResultSet rs = qexec.execSelect() ;

// The order of results is undefined.
for ( ; rs.hasNext() ; ) {
QuerySolution rb = rs.nextSolution() ;
// Get title - variable names do not include the â€™?â€™
RDFNode y = rb.get("individu");
Resource z = (Resource) rb.getResource("individu");
out.println(z.getLocalName());
out.println("<br>");
//listrang.add(y)}
}}finally {
// QueryExecution objects should be closed to free any system resources
}qexec.close() ;
out.println( "</center>");
out.println("</head>");
out.println("</html>");
}}
```

III.3 Présentation de l'application :

Notre application permet soit de visualiser le méta model et l'ontologie, consulter ou interroger la base de connaissances. Après exécution de l'application la fenêtre d'accueil suivante s'affichera.

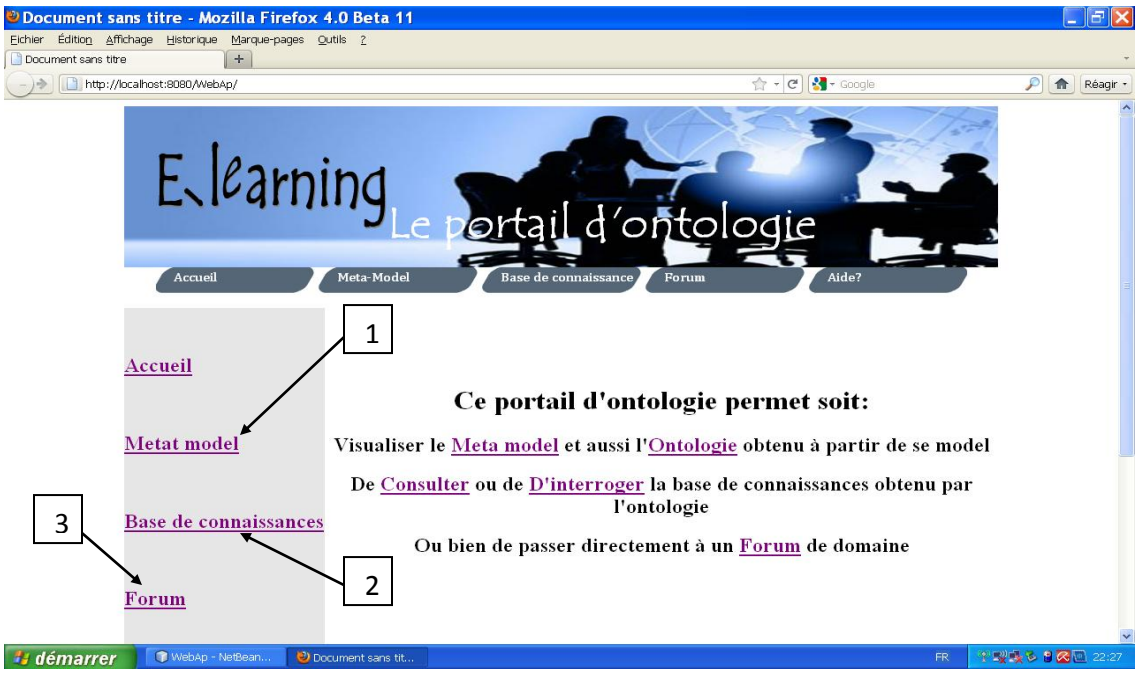


Figure25 : Page d'accueil de l'application.

1 : Lien pour afficher la hiérarchie de diagramme de classe.
2 : lien vers une autre fenêtre qui porte sur la base de connaissances.
3 : lien vers une adresse d'un forum sur internet.
Si nous cliquons sur le lien Meta model c'est la fenêtre suivante qui va s'afficher.

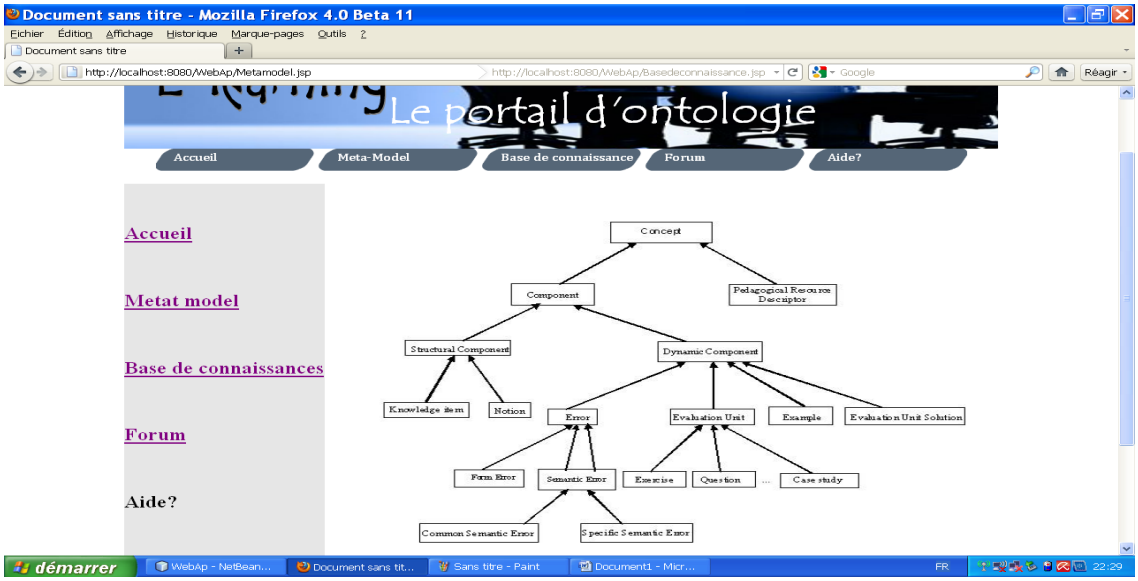


Figure26 : Page de description du méta model.

Si nous cliquons sur le lien Base de connaissances c'est la fenêtre suivante qui s'affichera :

Figure27 : Fenêtre base de connaissance.

Si nous cliquons sur le lien consulter le fichier OWL y-aura consultation du fichier OWL de notre base de connaissances comme le montre la figure suivante :

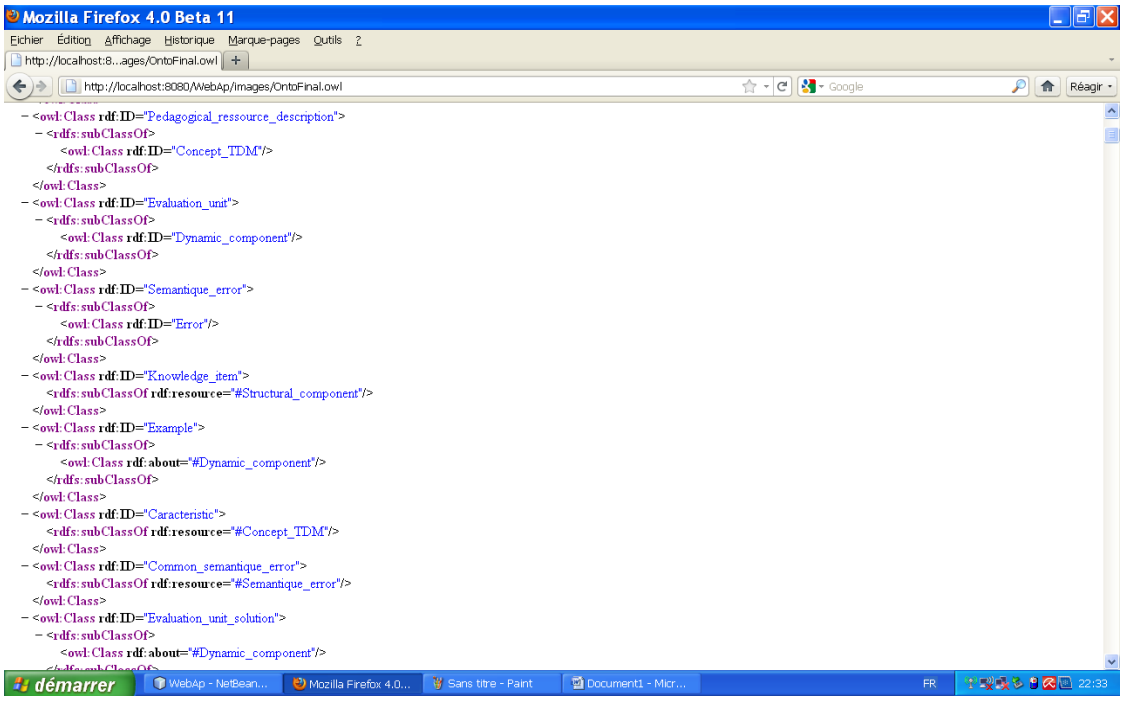


Figure28 : fenêtre d'affichage du fichier OWL

Si nous cliquons sur le lien interroger la base de connaissances et c'est la partie sur qui nous travail est concentré la fenêtre suivante s'affichera :

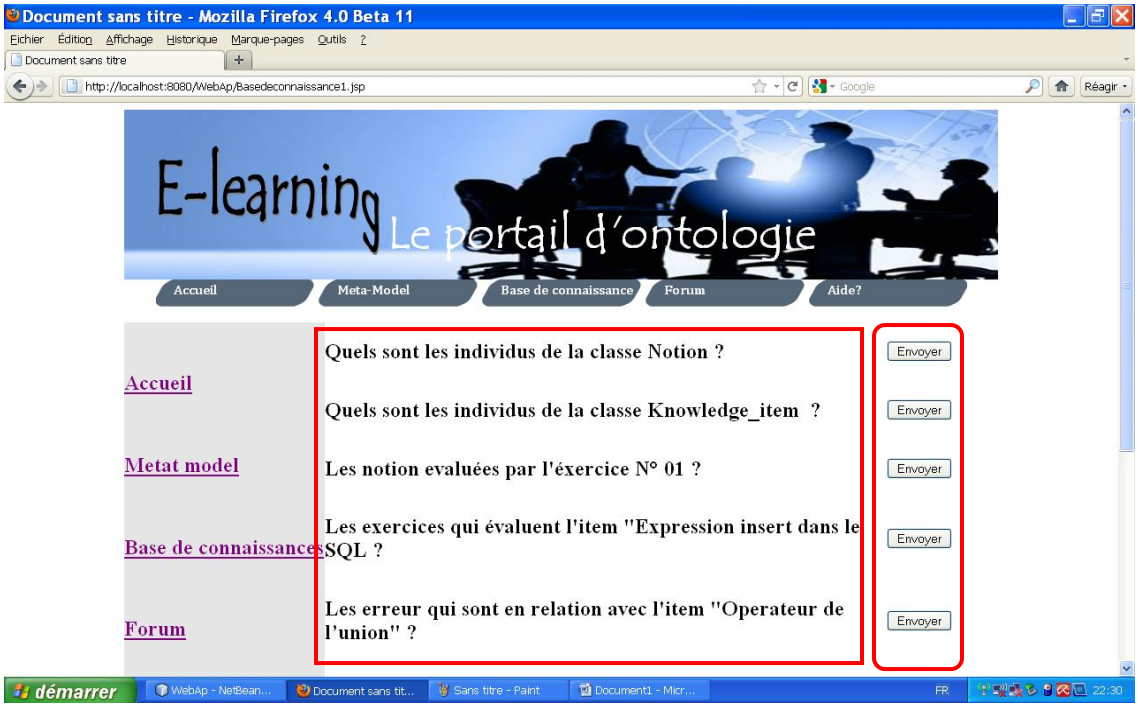


Figure29 : fenêtre d'interrogation de la base de connaissances.

Comme c'est visible sur la fenêtre y-a des questions et juste à coté des boutons. Nous choisissons une question ensuite nous cliquons sur le bouton correspondant. La question sera interpréter en requête SPARQL avec qui nous interrogerons le fichier OWL après une connexion à ce dernier. Les figures suivantes nous montrent quelques exemples de réponses.

Figure30 : réponse à "quels sont les individus de la classe Notion".

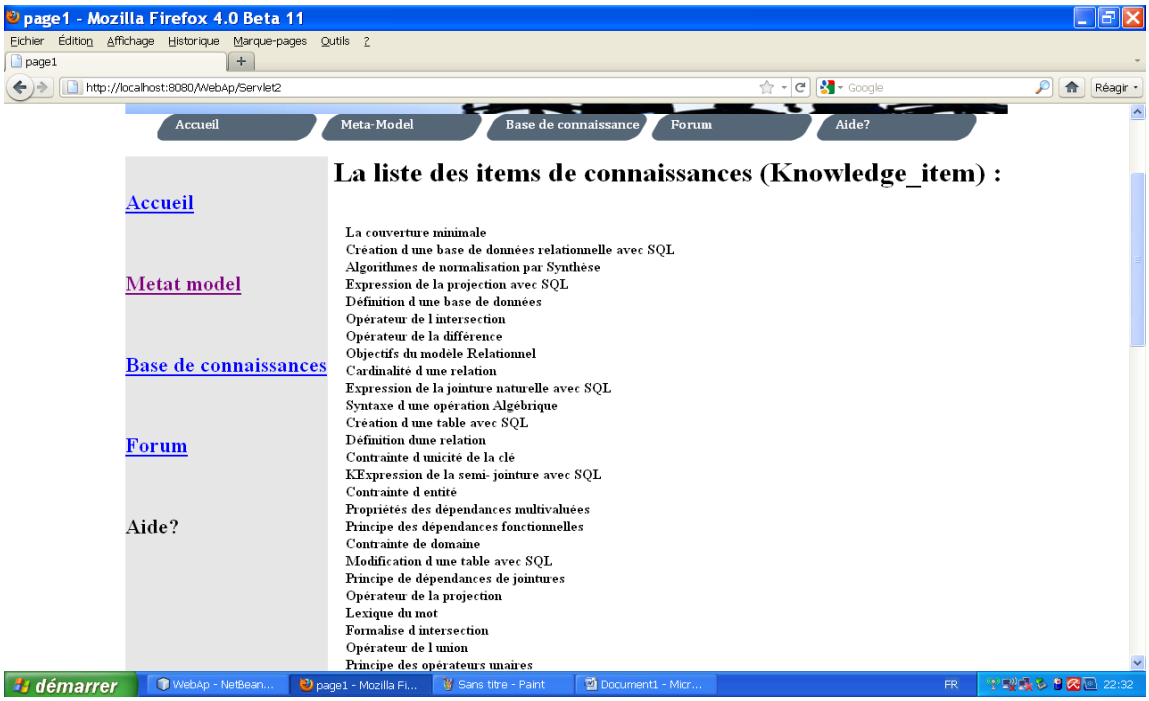


Figure31 : réponse à "quels sont les individus de la classe Knowledge item".

Figure32 : réponse à la question « les exercices qui évaluent l'item "Expression_insert_avec_SQL" ».

Conclusion :

Dans ce présent chapitre nous avons vu les outils utilisés pour l'exploitation de notre ontologie ensuite nous avons présenté l'exploitation elle-même dans ces deux cas : protégé et java en décrivant les requêtes utilisées et quelques interfaces de notre application.

Conclusion générale

Conclusion générale :

La puissance des applications qui utilisent des ontologies laisse à penser que leur place au sein des systèmes de E-Learning ne peut que croître. Cette puissance vient du fait que les systèmes qui utilisent les ontologies font des traitements sémantiques sur des grandes quantités d'information, réduisant ainsi le taux de bruit et de silence des réponses obtenues.

Le travail envisagé consiste à étudier les ontologies de domaine d'enseignement pour un web sémantique destiné pou l'e-Learning de point de vue applicatif.

Pour se faire nous avons opérationnalisé une ontologie de domaine d'enseignement déjà conçue en la traduisant dans un des langages d'ontologie OWL et SWRL pour ces différentes règles d'inférences en utilisant les outils de web sémantique, en effet, nous avons utilisé l'éditeur d'ontologie **Protégé** pour l'édition de l'ontologie, des règles et pour les manipuler avec le langage SPAQL.

D'autre part, nous avons utilisé l'API Jena pour l'exploitation de l'ontologie dans un environnement Java.

Parmi les perspectives de recherche que nous avons déduite de notre travail, nous avons:

- Possibilité et intérêts de raisonnement offert par le langage SWRL.
- La conception et le développement d'outils d'annotations automatiques qui intègrent les aspects sémantiques.
- Quel est l'apport du web sémantique pour améliorer les système de gestion de base de données, ou bien l'apport des base de données pour la technologie web sémantique.
- Expérimenter avec des cas concrets de système e-Learning l'apport du web sémantique.
- Les services requis pour les applications de e-Learning, en particulier pour le stockage des ressources, varient selon les applications de e-Learning. Dès lors il

est nécessaire de définir des entrepôts (structures) de métadonnées modulaires capables d'installer uniquement les services demandés.

- Comment exploiter les ontologies OWL dans les environnements PHP.

Références bibliographiques

Références bibliographique :

[BAC, 06] : BACH Thành Lê, Construction d'un Web sémantique multi-points de vue,DOCTEUR EN SCIENCES, spécialité informatique, L'École des Mines de Paris à Sophia Antipolis, 2006

[BEN, 09]: Benjamin Harbelot – Emmanuel Neret – Yoan Chabot
IUT Informatique de Dijon "convertisseur UML vers rdf/rdfs" 2009

[BER, 01]: Berners-Lee, T., Hendler, J., Lassila,O « The semantic Web » Scientific American, May 2001, p.28 -37

[BOU, 07] : BOUGCHICHE Lilia Ecole Doctorale STIC 2007
VERS une ontologie pour le dispositif d'interaction

[BOU, 04] Web Sémantique et e-Learning . Sabri Boutemedjet. Cours IFT6261. 2004.

[CHA, 98] : B. Chandrasekaran , J. R. Josephson and V. R. Benjamins, "Ontology
of Tasks and Methods," (Acrobat), 1998 Banff Knowledge Acquisition
Workshop.

[DIE, 01] DIENG R., CORBY O, GANDON F., GIBOIN A., GOLEBIOWSKA J., MATTA N. & RIBIÈRE M., « Méthodes et outils pour la gestion des connaissances : une approche pluridisciplinaire du knowledge Management ». Dunod, 2 edition. 2001.

[ABE, 04]: Abel M.-H. Utilisation de normes et standards dans le projet MEMORAe, In Distances et savoirs. 2004.

[FUR, 04]: FÜRST F., « Contribution à l'ingénierie des ontologies : une méthode et un outil d'opérationalisation ». PhD thesis, École Polytechnique de l'Université de Nantes (EPUN), Nantes, France. 2004.

[GRU, 93]: GRUBER T., " A translation approch to portable ontology specifications". Knowledge Acquisition, 5(2),. 1993.

[GUA, 98] : N. Guarino. "Formal ontologies and information systems". In N. Guarino, editor, Proceedings of FOIS'98, IOS Press, Amsterdam, 1998

[GUC, 94] : GUARINO N., CARRARA M. & GIARETTA P., "Formalizing ontological commitments". In National Conference of the American Association on Artificial Intelligence (AAAI). 1994

[HAC, 09] : Fakhr-eddine HACHEMI Descript ion sémant ique des objets d'apprent issage à base de modèles de contenu Mgister en informatique 2009. Ecole doctorale : S.T. I .C

[KAS, 02] : G. KASSEL, « Ontospec : une méthode de spécification semi-informelle d'ontologies ». In Actes des journées francophones d'Ingénierie des Connaissances (IC'2002).

[KAV, 04] : Kaveh BAZARGAN. Mémoire de DEA en Management et Technologies des Systèmes d'Information (MATIS). Groupe Interface des Systèmes d'Information (ISI) Centre Universitaire d'Informatique (CUI) Université de Genève Suisse. 2004

[KMS, 04]: Walid Kassem, Ahmad Mounajed, Nadia Saadoun « Etat de l'Art du E- Learning », Projet soutenu le 16/02/2004 Université de Paris 2, 2004.

[LAU, 02] : Philippe Laublet, Chantal Reynaud, Jean Charlet « Sur quelques aspects du Web sémantique » 2002.

[MAM, 07]: MAHIDDINE Mehanna & MISSOUM Mehenna, Conception et réalisation d'une ontologie dans le domaine des hydrocarbures pour la recherche d'information, I.N.I 2007.

[MBB, 09]: "Consultation d'ontologies OWL-DL au travers de SPARQL" Cours, Uni- versité Montpellier II, 2009. [63] Claude Moulin, Fathia Bettahar, Jean-Paul Barthès

[PEL, 04]: GÓMEZ-PÉREZ A., FERNANDEZ-LOPEZ M. & CORCHO O., " Ontological Engineering". Advanced Information and Knowledge Processing. Madrid, Spain : Springer, 2 edition. 2004.

[POU, 09]: Servlets et JSP de philippe POULARD 2009.

[RKA, 09]: RABHALAH Kahina ET KERMOUN Lynda "Prise en charge de réponses d'apprenant sous forme d'arbres algébriques en E-learning des bases de

données relationnelle". INGENIEUR (UMMTO), 2009.

[SEB, 01] : Sébastien George « apprentissage collectif à distance. SPLACH : un environnement informatique support d'une pédagogie de projet », Thèse de doctorat de l'université du Maine, soutenu le 11 juillet 2001

[SOW, 00]: (ICCS'2000) SOWA J., "Ontology, metadata and semiotics". In 8th International Conference on Conceptual Structures 2000.

[STA, 00] : S. Staab, A. Maedche. «Axioms are objects too: Ontology engineering beyond the modeling of concepts and relations.» Research report 399, Institute AIFB, Karlsruhe, 2000.

[STO, 01] : Stojanovic L., Staab S., Studer R. (2001) "eLearning based on the Semantic Web", In WebNet2001, World Conference on the WWW and Internet, October 23-27, 2001, Orlando, Florida - USA.

[these doctorat 2010]: Thèse doctorat de madame BOUARAB Farida 2010

[USG, 96]: Uschold M. et Grüninger M.,"ONTOLOGIES: Principles, Methods and applications". Knowledge engineering review, vol.11, N.2, 1996.

[USK, 95] : M. USCHOLD et M. KING, "Towards a Methodology for Building Ontologies". In Proceedingsof the Workshop on Basic Ontological Issues in Knowledge Sharing at the International JointConference on Artificial Intelligence (IJCAI'1995), 1995.

Webiographies :

[01] : http://www.w3.org/TR/2004/REC-owl-features/.

[02] : http://cyberzoide.developpez.com/graphviz/#LII

[03] : http://fr.wikipedia.org/wiki/NetBeans

[04] : http://jena.sourceforge.net/

[05] : http://fr.wikipedia.org/wiki/ Dreamweaver]

[06] : http://fr.wikipedia.org/wiki/Html

.