Laurent CARDONER
IUP ISI 3ème année

Rapport de Stage
1er avril – 31 août 04

Université
Paul
Sabatier
TOULOUSE III

IRIT
118 route de Narbonne
31062 TOULOUSE cedex 04

Philosophie, dis-moi des élégies.

Mylène Farmer, C'est une belle journée

Remerciements

Je tiens d'abord à remercier le directeur de l'IRIT, M. Luis Fariñas del Cerro, qui a bien voulu m'accueillir à l'IRIT.

Je remercie ma maîtresse de stage, Mme Josiane Mothe, pour m'avoir donné la possibilité de participer à un stage enrichissant et intéressant, ainsi que M. Jean-Paul Bodeveix, mon tuteur de stage.

Je remercie également Nathalie Hernandez qui m'a suivie tout le long du stage, et m'a ainsi apporté son aide, ses explications et a été patiente dans mon travail.

Je remercie Asma Brini, Nathalie Hernandez et Karen Sauvagnat pour la bonne humeur et la bonne ambiance qu'elles ont pu apporter dans le bureau.

Merci à toute l'équipe SIG.

Résumé

Dans le cadre de ma troisième année d'IUP ISI, j'ai effectué un stage d'une durée de cinq mois au sein de l'équipe Systèmes d'Informations Généralisés (SIG) de l'Institut de Recherche en Informatique de Toulouse (IRIT).

L'objectif a été de créer un logiciel capable de visualiser des ontologies, de calculer leurs mesures d'adéquation vis-à-vis d'un corpus de documents et de comparer deux ontologies (par la visualisation simultanée de deux ontologies). Pour se faire, j'ai d'abord tester et comparer des outils de visualisation d'ontologies existants sous la forme de plug-ins du logiciel d'éditeur d'ontologies *Protégé*. Au vu des résultats de cette analyse, il a été décidé avec Nathalie Hernandez, et après accord de Josiane Mothe, de développer une nouvelle application capable de répondre aux besoins exprimés.

J'ai donc d'abord défini l'architecture de la future application, répartie en quatre couches. Puis dans chaque couche, j'ai défini les paquetages présents. Pendant le développement, il m'a fallu trouver un algorithme d'affichage de graphe, en tenant compte du fait que l'application puisse afficher de grosses ontologies. Et pour finir, j'ai intégré les classes permettant le calcul des pouvoirs et entrepris leur visualisation.

Au final, ce stage m'aura permis d'apprendre et d'utiliser de nouveaux langages et outils. Il m'a aussi fait découvrir le travail d'une équipe, avec ses réunions et ses collaborations. Un stage donc enrichissant, tant sur le plan professionnel que relationnel.

Abstract

Within the framework of my third year of IUP ISI, I carried out a five months duration training period within the SIG team of the Institute of Research in Data processing of Toulouse (IRIT).

The objective was to create software able to visualize ontologies, to compute their powers in relation to a document corpus and then to compare two ontologies (with simultaneous visualization of two ontologies). For that, I first tested and compared visualization tools of ontologies which exist in the form of plug-ins of the ontologies editor software called *Protégé*. After this analysis, it was decided with Nathalie Hernandez, and after consent of Josiane Mothe, to develop a new application able to correspond to expressed needs.

So I first defined the architecture of the future application, divided into four layers. Then into each layer, I defined the packages. During the development, I had to find an algorithm to show a graph, taking the fact that the application has to show big ontologies into account. And the to finish, I integrated classes which compute powers and visualize them.

Finally, this training period allowed me to learn and use new languages and tools. It made me also discover teamwork, with meetings and collaborations. This training period was enriching, as well as professionally as in a relational way.

Sommaire

Présentation de l'IRIT

L'Institut de Recherche en Informatique de Toulouse (IRIT, UMR 5505 du CNRS) a été créé en janvier 1990 par son directeur fondateur Jean Vignolle au sein de Toulouse, qui a participé dès la fin des années cinquante au développement de l'informatique en France, et qui est l'un des tous premiers pôles nationaux et européens dans ce domaine.

L'IRIT résulte de la fusion de trois laboratoires :

- C.E.R.F.I.A. (Cybernétique des Entreprises, Reconnaissance des Formes, Intelligence Artificielle)
- C.I.T. (Centre d'Informatique de Toulouse)
- L.S.I. (Langage et Systèmes Informatiques)

Aujourd'hui, l'IRIT occupe dans le domaine informatique une place centrale et rassemble plus de quatre cents chercheurs, enseignants-chercheurs et doctorants du Centre National de la Recherche Scientifique (CNRS), de l'Institut National Polytechnique de Toulouse (INP) et de l'Université Paul Sabatier (UPS). Son directeur actuel (depuis 1999) est M. Luis Fariñas del Cerro.

La recherche

Les travaux de recherches qui y sont menés sont regroupés en sept thèmes :

⇨ **Analyse et Synthèse de l'Information**

Ce thème concerne le traitement des images, des sons et des textes, tant dans le but de reconnaître et de comprendre leur constitution et leur contenu que dans le but de les recréer à partir de modèles.

- La **vision par ordinateur** vise à déterminer la projection géométrique liant une scène à son image pour la reconstruction de la scène ou pour simuler de nouvelles prises de vues à partir d'images données.
- La **synthèse d'images** étudie la modélisation de formes et de scènes, les algorithmes de visualisation et de rendu des propriétés optiques des objets.
- L'**informatique linguistique** étudie la modélisation des phénomènes linguistiques dans des systèmes automatiques du traitement du langage naturel et parmi les applications, l'indexation automatique et la recherche d'informations dans des textes.

- Le **traitement de la parole** inclut la reconnaissance de la parole, l'identification du locuteur, l'identification de la langue en utilisant une approche statistique du traitement du signal.

⇨ **Indexation, Recherche et Stockage d'Informations**

- **Indexation et recherche d'information**

  L'objectif de ce thème est d'aborder les problématiques liées à la prise en compte des données atypiques (données textuelles, multimédia, semi-structurées, multidimensionnelles) dans les bases de données, qui s'éloignent du modèle traditionnel classiquement supporté par les SGBD - requêtes précises sur des données exactes - pour aller vers des besoins d'exploitation fine et rapide de larges volumes de données à structure complexe ou inconnue.

- **Indexation multimédia**

  Ce thème a pour objectif de développer des modalités d'accès dites 'intelligentes', par automatisation de la tâche d'indexation, dans le cadre d'applications diverses dans des domaines tels que les télécommunications, l'éducation, l'expertise, l'archivage, la télévision, le cinéma... que ce soit pour des applications grand public ou professionnelles, afin de traiter beaucoup plus de requêtes et d'alléger les traitements extrêmement fastidieux qu'engendre l'exploitation des volumes de données numériques actuellement disponibles (librairies audio sur Internet, bases de données, bouquets numériques,...) et dont l'exploitation est en grande partie conditionnée par leur facilité d'accès.

⇨ **Interaction, Autonomie, Dialogue et Coopération**

Ce thème concerne une population de chercheurs de disciplines diverses dont la préoccupation commune concerne les relations entre agents, que ceux-ci soient de même nature ou de nature différente (par exemple les communications homme-machine orale et multimodale).

- **Modélisation**
- **Systèmes formels pour le dialogue et l'interaction**
- **Systèmes auto-adaptatifs émergents**
- **Ergonomie cognitive**

⇨ **Raisonnement et Décision**

Huit axes de recherches sont développés :

- **Raisonnements plausibles**
- **Raisonnement sur le changement**
- **Raisonnement sur l'espace et le mouvement**
- **Argumentation et formalisation de l'interaction entre agents**
- **Décision et planification**
- **Apprentissage**
- **Méthodes et algorithmes**
- **Représentation, raisonnement, langue et cognition**

⇨ **Modélisation, Algorithmes et Calcul Haute-Performance**

Trois axes de recherche sont développés :

- **Algèbre linéaire numérique et grid computing**
- **Optimisation et contrôle**
- **Algorithmes parallèles asynchrones**

⇨ **Architecture, Systèmes et Réseaux**

Ce thème regroupe essentiellement les équipes intéressées par les supports d'exécution et de communication indispensables au traitement de l'information. On trouve ainsi :

- **Architecture**
- **Systèmes temps réel et embarqués**
- **Ingénierie des réseaux et télécommunications**
- **Réseaux et services**
- **Optimisation dynamique de requêtes parallèles réparties**

⇨ **Sûreté de Développement du Logiciel**

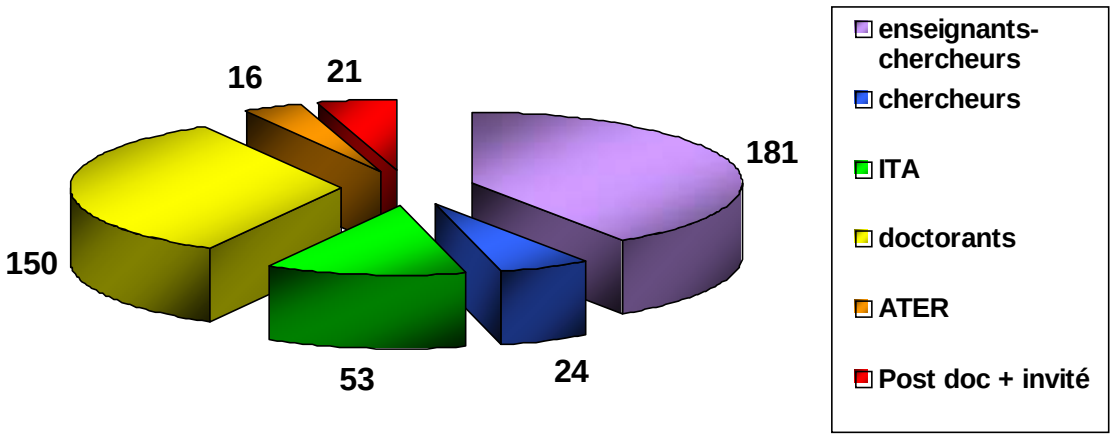
Dans le domaine de l'aide au développement de logiciel, ce thème s'intéresse plus particulièrement à conduire des activités de recherche qui visent à réduire les coûts et les délais de validation des logiciels, par l'utilisation de méthodes formelles.

Ces recherches, à la fois théoriques et appliquées, sont synthétisées dans les trois axes suivants :

- **Théorie des types et assistance à la preuve**
- **Ingénierie des systèmes parallèles et répartis**
- **Aide au développement de composants logiciels**

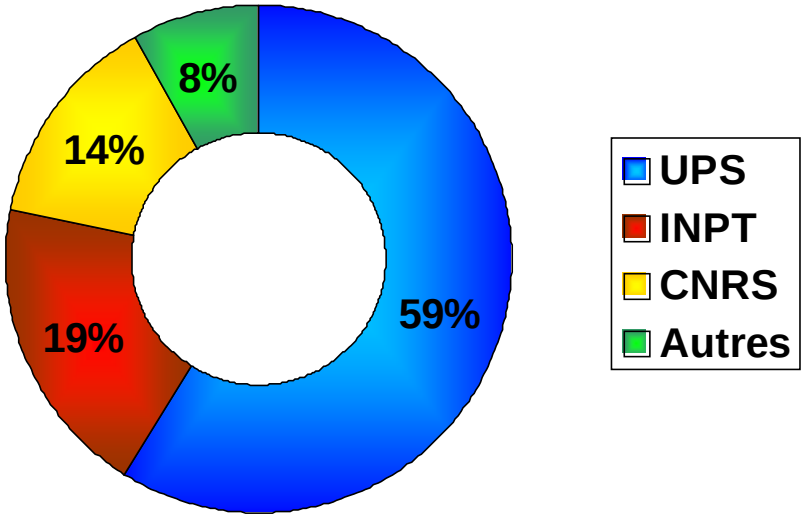
Le personnel

L'effectif de l'IRIT est de 445 personnes réparties comme suit :



ITA : Ingénieurs, Techniciens, Administratif
ATER : Attachés Temporaires d'Enseignement et de Recherche

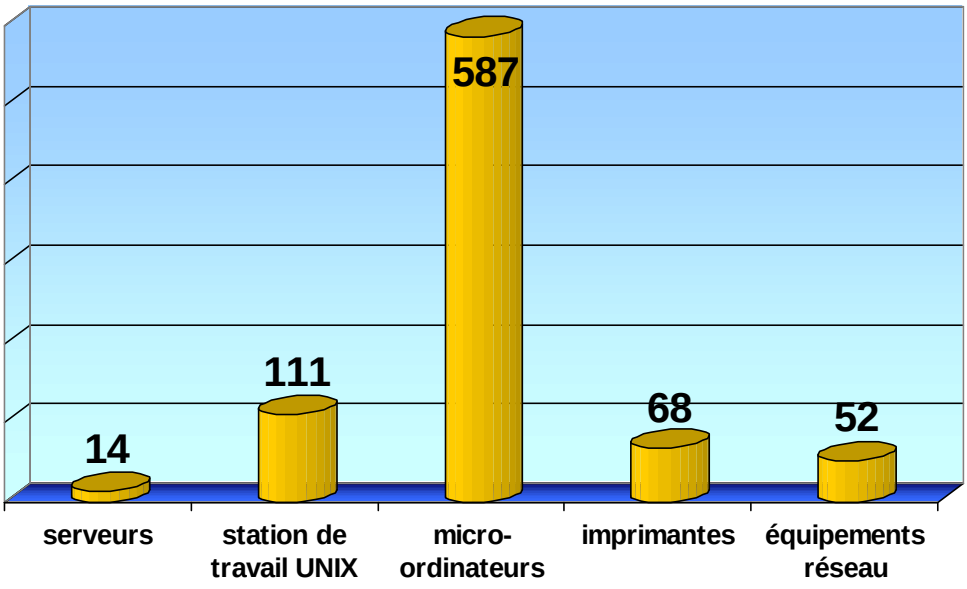
Personnel 2004



Ressources humaines 2001

Les équipements

Le parc des machines de l'IRIT est un ensemble hétérogène (Macintosh, PC, Station…) en perpétuelle croissance. Aujourd'hui, il est composé de 832 équipements (voir le diagramme de répartition ci-après) interconnectés au travers d'un réseau local, basé sur les protocoles TCP/IP, Ethernet, ATM.



Parc des machines 2004

Les services

L'IRIT comprend six services :

- **centre informatique** qui soutient et appuie les travaux de recherche de l'Institut. Deux entités le composent :
  1. le service informatique responsable de l'exploitation des moyens généraux et de la définition de la politique d'équipement
  2. la cellule recherche et développement qui participe aux activités de recherche

- **centre de documentation, bibliothèque**

- **service de valorisation et relations industrielles** dont la mission est de valoriser les résultats de la recherche et de transférer les technologies, le savoir-faire et les connaissances au niveau régional, national et international

- **service communication**

- **service technique** (audiovisuel, reprographie, …)

- **service administratif et financier**

L'équipe Systèmes d'Informations Généralisés (SIG)

C'est au sein de cette équipe que j'ai effectué mon stage. Elle fait partie du thème *Indexation, Recherche et Stockage d'Informations*.

Les recherches menées au sein de l'équipe SIG s'articulent actuellement autour de concepts de Base d'Informations selon deux approches complémentaires permettant d'appréhender :

- les gisements d'informations et bases de données multidimensionnelles, dont l'objectif est de proposer, développer et expérimenter des techniques et stratégies d'identification de sources d'information, de mémorisation, de filtrage/recherche d'information (explicite ou cachée) et de présentation, et de coordonner ces opérations à travers un plan stratégique global.
- les systèmes d'information et d'ingénierie documentaire pour proposer, développer et expérimenter des modèles, des langages, des méthodes et des techniques autour du concept de bases d'objets documentaires ou hyperbases. Cette approche privilégie l'élicitation et la manipulation de structures irrégulières issues des informations manipulées, via des langages de type SQL, OQL, XmlQL,...

Ces deux axes reposent sur trois thèmes :

① ***Filtrage, Recherche, Exploration d'Informations***

Ce thème concerne la mise au point de SRI (Systèmes de Recherche d'Informations) sophistiqués orientés bases textuelles avec la prise en compte du concept de profil (profil utilisateur, profil d'usage), l'intégration de possibilités d'interrogation multilingue (combinant par exemple Français, Anglais, Allemand,...), la reformulation automatique de requêtes par réinjection d'informations prenant en compte des préférences utilisateurs ou extraites via des outils d'analyses multidimensionnelles.

Ce thème repose sur deux composantes :

❶ ***SIG/RFI (Recherche et Filtrage d'Informations)***

Les travaux menés à l'heure actuelle s'inscrivent dans le domaine de la recherche d'informations complexes dans un environnement d'applications hétérogènes. Plus précisément, ils abordent plusieurs thèmes complémentaires relatifs à la représentation, la recherche, le filtrage et l'analyse dans des corpus de documents.

❷ ***SIG/EVI (Extraction et Visualisation d'Informations)***

La Découverte de Connaissances à partir de bases de données a été définie comme l'extraction à partir de données d'information implicite, non connue a priori mais utile. De façon similaire, la découverte d'informations a pour objectif d'extraire, à partir d'informations textuelles, des informations cachées ou des modèles. Parmi les problématiques de recherche sous-jacentes on peut citer : les modèles de représentation de l'information en vue de son analyse, les méthodes d'analyse exploratoire des données, les interfaces de visualisation, l'adaptation aux besoins des utilisateurs. Les travaux de recherche s'orientent dans ces différents domaines.

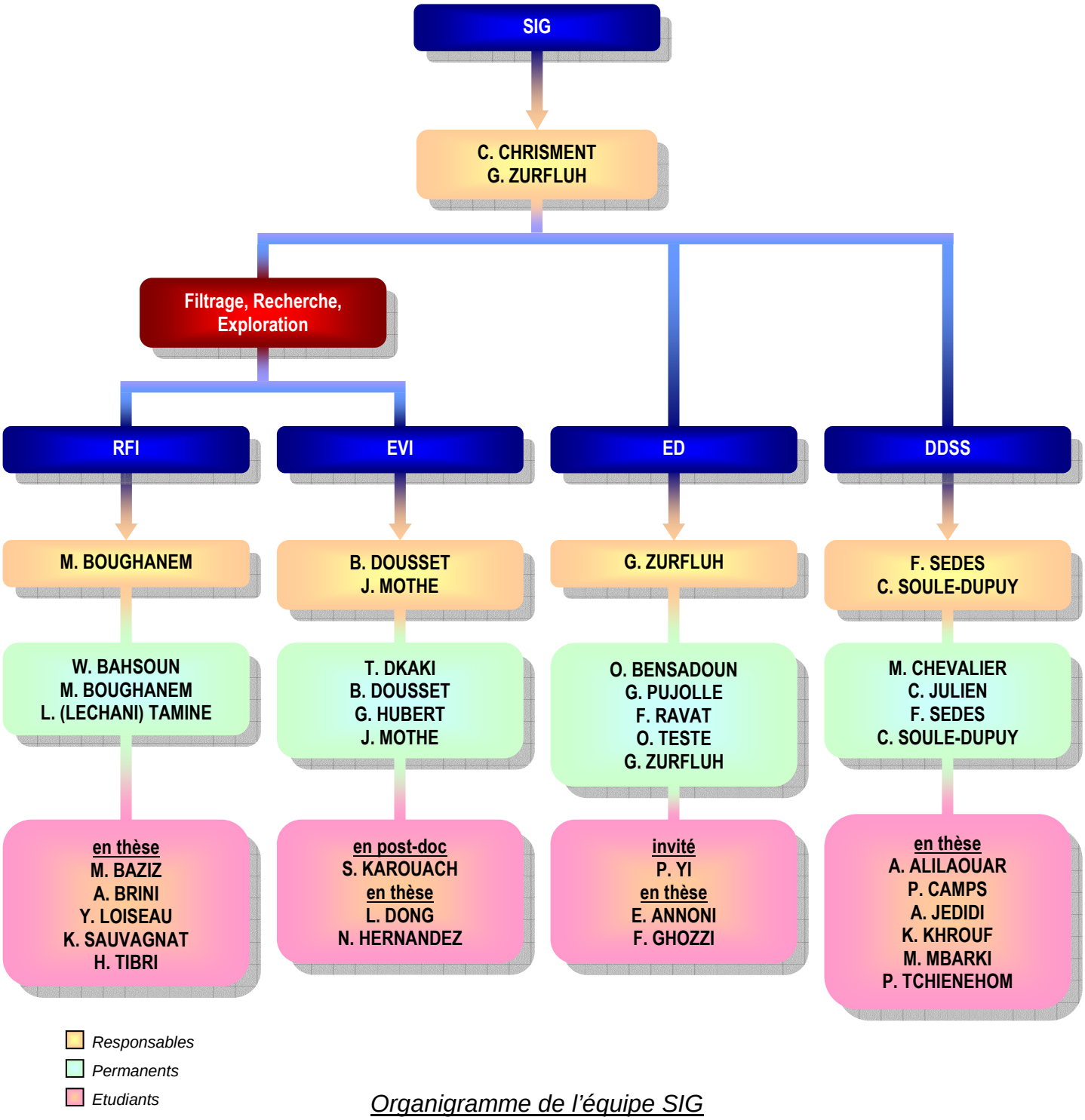
② ***Entrepôts de données (SIG/ED)***

Un entrepôt de données stocke des données utiles aux décideurs pour effectuer diverses analyses économiques grâce aux techniques de type On-Line Analytical Processing (OLAP) ou fouille de données (Data Mining). L'étude des modèles de description des entrepôts de données et des méta-données permet de caractériser les données issues de sources hétérogènes et de prendre en compte leur historisation à différents niveaux de granularité. On apporte ainsi une extension aux concepts présents dans les Bases de Données Temporelles, on y appréhende les aspects Méthodologie de Conception d'entrepôts et de manipulation de données multidimensionnelles.

③ ***Documents, Données Semi Structurées et usages (SIG/DDSS)***

Actuellement, le concept de méta données (spécifiées à priori, extraites ou générées) permet de décrire par média les ensembles d'informations constituant une base de données multimédia (description par facettes, par annotations qui

peuvent être hiérarchisées, stratifiées, multidimensionnelles). A partir de ces descriptions orientées média, on propose une démarche d'unification pour en proposer une vision générique et permettre un traitement uniforme des requêtes qui vont combiner ces médias.



Organigramme de l'équipe SIG

Présentation du projet

Etat des lieux

Les systèmes de recherche d'information visent à restituer les documents pertinents par rapport à un besoin en information. Ils se basent sur une représentation des documents pour permettre leur organisation et recherche ultérieure. Cette représentation, appelée aussi indexation, utilise le contenu des documents et est réalisée soit manuellement, soit automatiquement.

- **L'indexation manuelle** est souvent choisie lorsque les corpus de documents sont spécialisés. Dans ce cas, elle se base sur l'utilisation d'un langage commun (ontologie ou représentation du domaine). Les indexeurs analysent le contenu du document à indexer et choisissent les « entrées » de l'ontologie qui serviront de clés d'accès aux documents pour leur recherche.

- **L'indexation automatique** peut également faire appel à une ontologie, et dans ce cas l'indexation s'apparente à de la catégorisation de documents par rapport aux entrées définies dans celui-ci.

L'utilisation d'une représentation du domaine via son vocabulaire à l'avantage d'atténuer la subjectivité du choix de termes représentatifs pour un texte (connaissance du domaine, synonymie, etc.). Cependant, l'utilisation d'une telle ressource implique son adéquation avec les documents à indexer. En particulier, sa mise à jour est nécessaire pour tenir compte de l'évolution du vocabulaire du domaine.

Sujet du stage

Masse de données en Astronomie : aide à l'extraction et à l'utilisation d'une ontologie du domaine

L'objectif de ce stage est de développer des outils servant à déterminer l'adéquation du vocabulaire commun avec les textes à indexer à partir d'une analyse statistique et sémantique de leur contenu.

Plusieurs ontologies d'un même domaine sont souvent disponibles. Des mesures ont été définies dans [Hernandez 2004] pour déterminer quelle ontologie est la plus adaptée pour indexer une collection documentaire. L'objectif de ce stage est de développer une interface permettant de visualiser le résultat de ces mesures sur plusieurs ontologies. L'interface devra présenter les mesures obtenues pour les différentes « entrées » des différentes ontologies ainsi que le lien entre les entrées et leurs apparitions dans les documents. Une première étape consistera à se familiariser avec le format OWL (standard de représentation des ontologies reconnu par le w3c). Le standard OWL sera utilisé pour représenter les ontologies visualisées dans l'interface. Une deuxième étape sera de concevoir une interface pour que ce format soit pris en compte et que les valeurs des mesures y soient représentées.

Le cadre applicatif de ce stage est lié au projet MDA Masse de Données en Astronomie, Action Concertée Incitatives du département STIC du CNRS (http://www.technologie.gouv.fr/recherche/fns/mdonnees.htm). Ce travail est également lié au projet iAstro (www.iAstro.org). Plus généralement, ce projet vis à proposer des mécanismes qui pourraient s'avérer indispensables dans le cadre du Web sémantique (nouvelle génération du Web actuel) et de ces applications concrètes.

Contraintes

Ce projet s'appuiera en particulier sur :

- les normes et standards du World Wide Web Consortium (OWL, XML),
- l'ontologie IAU (http://msowww.anu.edu.au/library/thesaurus/)
- les outils d'analyse statistique et de traitement automatique des langues déjà développés à l'IRIT ou en collaboration avec d'autres laboratoires (Tétralogie, Syntex, Caméléon, TERMINAE)
- les ressources issues de ADS (http://adswww.harvard.edu/) « The NASA Astrophysics Data System », serveur de données dans le domaine

Les langages de programmation utilisés seront le Java et le Perl (langage adapté aux traitements des documents textuels).

Les concepts de base

Le Web sémantique

Le Web actuel est un espace de stockage de documents permettant de lier les ressources par des liens de structure. Il peut en effet être considéré comme une librairie digitale reliant les documents par des liens hypermédia mais aussi comme une gigantesque base de données offrant de plus une plateforme d'applications. Il forme un portail commun aux applications accessibles par des pages Web et permet de présenter leurs résultats sous le même format de page web. Il représente également une plateforme multimédia sous laquelle il est possible d'échanger de nombreux formats d'information. Il définit également un schéma nommé permettant d'identifier de façon unique les documents. L'ensemble de ces caractéristiques ne répondent cependant que partiellement aux objectifs fixés lors de la création du Web.

L'expression « Web sémantique » a été introduite par Tim Bernes-Lee, fondateur et président du consortium World Wide Web (W3C).

Elle fait référence à une vision du Web de demain comme étant un vaste espace d'échange de ressources entre les êtres humains et les machines, permettant une exploitation, qualitativement supérieur au Web d'aujourd'hui, de grands volumes d'informations.

Le Web sémantique est une infrastructure permettant l'utilisation de connaissances formalisées en plus du contenu informel actuel du Web. Elle a pour but d'atteindre de manière plus robuste et efficace les données présentes sur le Web en se basant sur la définition de consensus et de standards.

Pour comprendre la notion de « Web sémantique », il faut analyser le Web d'aujourd'hui et les nouveaux besoins de ses utilisateurs. Cela ne fait aucun doute, le Web est aujourd'hui une gigantesque source d'informations. C'est en tant que telle qu'il a été conçu. Pourtant, si ces informations sont compréhensibles par l'homme, elles ne le sont pas forcément par des logiciels parcourant le Web. En effet le format de données le plus courant actuellement est le HTML. Ce standard a le mérite d'être simple, mais en contrepartie il est quasi inefficace pour donner une dimension informationnelle supplémentaire aux données, c'est-à-dire exprimer des « méta-données », et ne supporte pas l'apparition de nouveaux services plus évolués que la simple recherche textuelle.

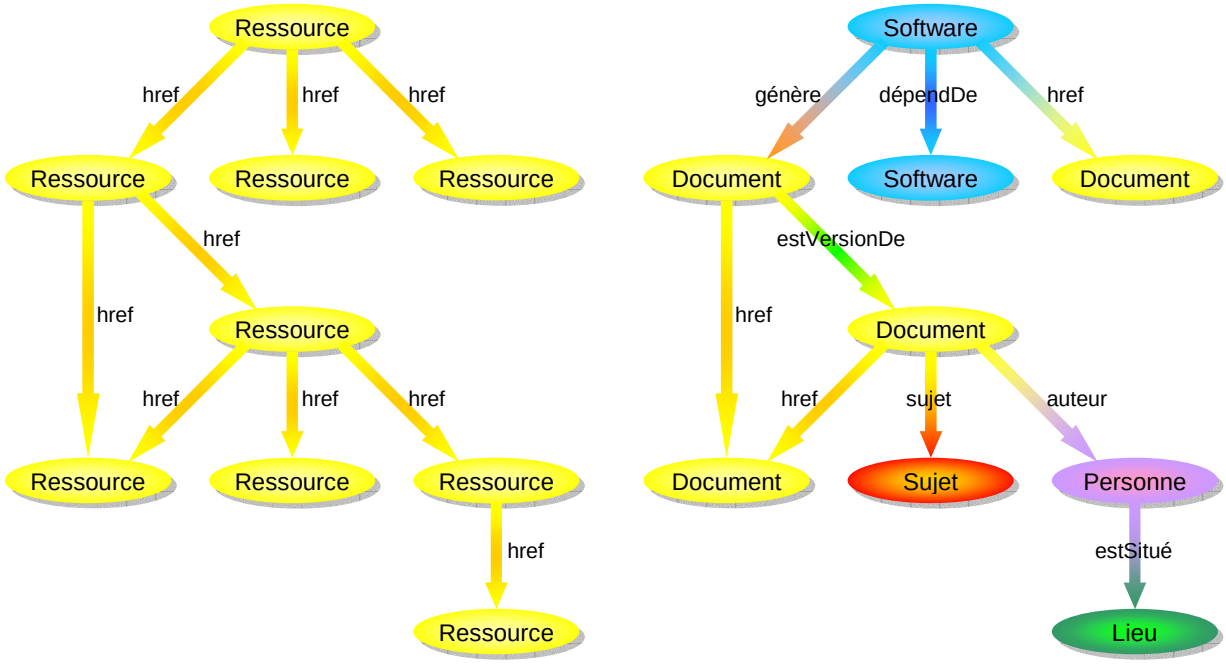
C'est pourtant une étape indispensable, que celle de l'expression de méta-données, pour permettre à un logiciel, tel qu'un moteur de recherche ou un navigateur, de s'affranchir des techniques de recherche brute de l'information utilisées actuellement.

En effet la recherche en texte intégral utilisée par les moteurs de recherche, comme Lycos, qui se distingue sous cet angle des autres annuaires, s'avère finalement peu efficace. Tandis que la recherche basée sur le parcours de répertoires de sites, typiquement la méthode de recherche utilisée par Yahoo, s'avère assez coûteuse ; en effet le catalogage des sites Web est réalisé manuellement et en limite donc sa portée (nombre de documents indexés). Ces deux méthodes restent donc insuffisantes pour fournir des réponses pertinentes aux requêtes des Internautes.

Quelques moteurs de recherche pourtant utilisent les balises <META> insérées dans le code source HTML du document. Il s'agit de la méthode la plus sûre d'indexation dans la mesure où le robot indexe les mots-clefs qui donnent une courte description du document, choisis par l'auteur du site lui-même (par exemple : date, auteur, etc. du document). C'est le début d'une recherche basée sur des méta-données. Le moteur de recherche d'Altavista par exemple utilise les balises <META> pour fournir ses réponses aux requêtes des utilisateurs.

L'intention du Web sémantique est de s'appuyer sur des descriptions structurées de l'information qu'il contient, plutôt que sur la structure textuelle du document. Ceci afin de construire des documents organisés et compréhensibles par des machines, par des agents qui pourront parcourir le Web de façon automatique, par des moteurs de recherche et d'autres services plus avancés.

Le schéma ci-dessous illustre l'importance des modifications apportées par le Web sémantique au niveau des liens unissant les ressources. Le contenu du Web d'aujourd'hui est lisible par des humains et par des machines mais il n'est compréhensible que pour les humains. Le Web sémantique vise à ajouter des annotations sémantiques, à propos des ressources du Web, qui soient compréhensibles et exploitables par des logiciels.



Le web actuel *Le web sémantique*

Le Web sémantique est une extension du Web actuel dans lequel l'information rajoutée a un sens bien défini, permettant ainsi aux ordinateurs et aux personnes de travailler en coopération. Les premières étapes pour tisser le Web sémantique dans la structure du Web existant sont déjà franchies (balises <META> par exemple).

La propriété essentielle du Web traditionnel est son universalité. La puissance des liens hypertextes est que n'importe quelles ressources peuvent être liées. L'information du Web varie selon plusieurs axes. L'un d'eux est la différence entre l'information produite pour la compréhension humaine et celle produite principalement pour les machines. D'un côté, des informations sont disponibles sur tous les sujets, du programme TV à la poésie et de l'autre, il existe des bases de données, des programmes et des sorties. Le but du Web sémantique est d'harmoniser cet ensemble hétéroclite.

Ainsi, le Web sémantique pourra fournir des services plus aboutis à ses utilisateurs (trouver l'information pertinente, sélectionner, localiser et activer le service nécessaire...). Il peut être vu comme une infrastructure complétant le contenu informel du Web actuel avec de la connaissance formalisée. Il peut conduire à faire cohabiter plusieurs degrés de formalisations allant de schémas de méta-données figés à des langages de représentation plus complexes.

Les ontologies

Dans le but d'ajouter de la sémantique au Web actuel, les ontologies sont utilisées comme mécanisme pour représenter de telles descriptions du domaine formel et partagé.

Définition

Le terme « ontologie », du grec οντοσ (être) et λογοσ (science), a été emprunté au domaine de la philosophie dans lequel il signifie « l'essence de l'essentiel ». Dans le domaine de la gestion de connaissance, le sens de ce mot est différent. La notion d'ontologie a d'abord été introduite comme « une spécification explicite d'une conceptualisation ». Cette définition a été légèrement modifiée par la suite. Une combinaison des deux définitions peut être résumée ainsi : « une spécification explicite et formelle d'une conceptualisation partagée ». Cette définition s'explique ainsi :

- ☑ *explicite* signifie que le « type des concepts et les contraintes sur leurs utilisations sont explicitement définies ».
- ☑ *formelle* se réfère au fait que la spécification doit être lisible par une machine.

- ☑ *partagée* se rapporte à la notion selon laquelle une ontologie « capture la connaissance consensuelle, qui n'est pas propre à un individu mais validée par un groupe ».
- ☑ *conceptualisation* se réfère à « un modèle abstrait d'un certain phénomène du monde basé sur l'identification des concepts pertinents de ce phénomène ».

Une ontologie fournit une base solide pour la communication entre les machines mais aussi entre humains et machines en définissant le sens des objets tout d'abord à travers les symboles (mots ou expressions) qui les désignent et les caractérisent et ensuite à travers une représentation structurée ou formelle de leur rôle dans le domaine. Différents niveaux sémiotiques sont à considérés dans une ontologie :

1. le niveau lexical recouvre l'ensemble des termes utilisés pour transcrire le sens des concepts.
2. le niveau conceptuel représente les concepts et les relations conceptuelles qui les relient.

De nombreux types de structure de connaissance se cachent derrière le mot ontologie (taxonomie, thesaurus,…). Ces structures de données peuvent être à la fois terminologiques (elles contiennent un ensemble de termes) et conceptuelles (elles définissent généralement des concepts). Cependant, ces structures peuvent différer par leur contenu (connaissances générales ou connaissances d'un domaine), par le type des relations sémantiques entre les concepts (relations taxonomiques, relations métonymiques, …), et par le niveau de formalisation (représentation logique, représentation dans un langage dédié aux ontologies…).

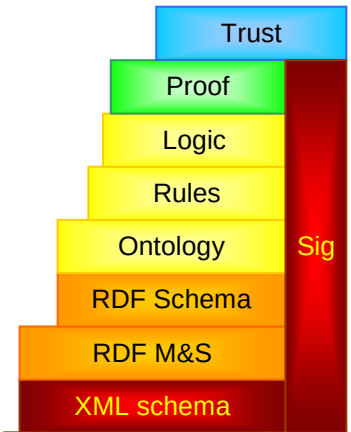
De plus, on distingue deux types d'ontologies :

① Les ontologies de tâches, qui organisent les méta-données.

② Les ontologies de domaines, qui représentent la connaissance liée au domaine.

Ontology Web Language

OWL est un nouveau langage d'ontologie créé pour être utilisée par les applications cherchant à traiter le contenu de l'information et non plus uniquement à présenter l'information. OWL se veut plus représentatifs du contenu du Web que XML, RDF et RDF schémas en apportant un nouveau vocabulaire avec une sémantique formelle.



OWL est une révision de DAML+OIL définit d'après l'expérience acquise lors de la création et l'utilisation de ce langage. OWL ajoute du vocabulaire pour décrire les propriétés et classes, par exemple la disjonction de classe, la cardinalité (exactement un), l'égalité, et des types et des caractéristiques (symétrie, transitivité) de propriétés plus riches.

OWL est basé sur 3 sous langages d'expressivité croissante : OWL lite, OWL DL, OWL Full :

- OWL Lite est fait pour des besoins préliminaires permettant de définir une hiérarchie et des contraintes simples. Il permet de définir facilement des thesauri ou taxonomies.
- OWL DL et Full sont basés sur OWL Lite auquel sont ajoutées des constructeurs supplémentaires.
  OWL permet de définir :

- ☑ Les classes : Les classes fournissent un mécanisme d'abstraction permettant de grouper des ressources partageant des caractéristiques similaires. Comme dans RDF, chaque classe OWL est associée à un ensemble d'individus appelés extension de la classe. Chaque individu de l'extension de la classe est une instance de la classe. Cependant, la notion de classe de OWL a pour but d'être plus expressive, elle a donc été redéfinie.
- ☑ Les propriétés : Une propriété est une relation binaire, s'appliquant soit entre une instance de classe et un littéral ou un type XML (premier type) ou soit entre les instances de deux classes (second type). Contrairement à RDFS, OWL permet de faire la différence entre ces deux cas de figures : DatatypeProperty permet de définir une propriété du premier type alors que ObjectProperty définit le second.

Utilisation des ontologies en Recherche d'Informations

Les systèmes de recherche d'information visent à restituer (tous) les documents pertinents (et seulement ceux là) par rapport à un besoin d'information exprimé par un utilisateur. Les systèmes d'exploration visent quant à eux à fournir de l'information élaborée à partir de l'analyse d'un ensemble d'informations relatives à un thème. Les index jouent un rôle primordial dans ces deux types de systèmes en définissant les descripteurs (mots ou groupements de mots) qui représentent le contenu des

documents et à partir desquels les documents peuvent être accédés ou analysés. L'indexation des documents n'est pas le seul moyen utilisé pour structurer et organiser une collection. Un modèle de structuration complémentaire consiste à utiliser les méta-données (des informations sur des informations) explicitement associées aux documents. Ces méta-données sont des données factuelles qui contiennent de l'information sur l'information comme par exemple le nom des auteurs, la date de publication, les mots clés choisis par les auteurs… Ces méta-données peuvent être combinées aux descripteurs issus de l'indexation pour fournir une représentation plus complète des documents de la collection. Cette représentation basée sur les méta-données et les descripteurs correspond à de la connaissance relative au corpus.

Quel que soit le mode de représentation interne des informations par le système, un des problèmes auquel l'utilisateur doit faire face est d'exprimer son besoin en information. Généralement quand un utilisateur sait ce que contient la collection, comment elle est structurée, ce qu'il recherche et comment il peut le décrire, il n'a pas de problème pour formuler sa requête. En fournissant les méta-données, le système peut aider l'utilisateur à évaluer le potentiel de la collection, cerner ses besoins et définir ses requêtes. Se pose cependant le problème de choisir quelles méta-données fournir à l'utilisateur et quelle organisation choisir pour représenter cette connaissance. Les ontologies semblent être la solution la plus adaptée à cette problématique.

Java

Historique

Les débuts de Java remontent à avril 1991, quand un petit groupe d'employés de Sun ont été déplacés dans un autre département pour travailler à un projet baptisé 'Green' voué à la réalisation d'applications électroniques commerciales. L'équipe Green était libre de faire ce qu'elle voulait. Sa première mission fut de créer un appareil capable de contrôler plusieurs équipements électroniques à la fois. Un langage fut créé à cette intention, le langage Oak, dont les premiers pas remontent à août 1991. Le nom Oak, trop proche d'une autre marque, fut rebaptisé Java en 1995. L'annonce officielle de Java et HotJava date, quant à elle, du salon SunWorld 1995. Le projet Green était originellement orienté vers la création de systèmes d'exploitation pour Personal Digital Assistant (PDA), set-top boxes, etc.

Le projet initial fut dissout et la technologie recentre vers le cd-rom, le multimédia en ligne et les environnements réseaux. Le maître d'œuvre de Java est James Gosling, qui commence par vouloir étendre le C++ avant de réaliser le premier compilateur Java écrit en langage C. Arthur Van Hoff implanta le compilateur Java en Java lui-même à la fin 1994.

Portabilité

L'avantage le plus important de Java est probablement sa portabilité, puisqu'il peut tourner sur n'importe quelle machine disposant d'un interpréteur Java. Cette

portabilité est fondamentale sur Internet, où un nombre important de machines et systèmes d'exploitation différents inondent le marché.

Dans un environnement interprété, le code Java n'est conçu pour aucune plate-forme spécifique mais est une sorte de code intermédiaire qui doit être traduit en code natif machine par un programme particulier faisant office de compilateur. Le code intermédiaire ou bytecode est conçu pour une machine qui n'existe pas, ou plutôt une Java Virtual Machine (JVM). Cela s'explique par la lenteur habituelle des interpréteurs, obligés de décoder chaque ligne de code avant leur exécution. Java pallie à cette lenteur par une sorte de première compilation générant du code plus rapide à convertir en code natif.

Cette portabilité va jusqu'à codifier une taille unique pour chaque type de données. Même si Windows 3.1 stocke ses entiers sur 16 bits, Windows 95 sur 32 ou un DEC Alpha sur 64 bits, un entier Java a la même taille mémoire dans toutes ses implémentations.

Strictement parlant, Java est un langage interprété. En effet, tous les codes de la JVM doivent être interprétés par chacune des plates-formes. L'interprétation s'opère via un navigateur World Wide Web ou un environnement runtime spécial.

Orienté objet

Le langage C++ est un compromis entre un langage C de troisième génération et un langage purement orienté objet comme Smalltalk. Même s'il s'inspire de la syntaxe du C++, Java a été réécrit à partir de zéro, de manière à être totalement orienté objet.

Java est un langage plus purement orienté objet que le C++, qui conserve de nombreux aspects de programmation procédurale puisqu'il n'est lui-même qu'une extension du langage C.

L'objet Java appartient à une classe. Cette classe est une collection de variables, de méthodes encapsulées dans un objet réutilisable et chargé dynamiquement en mémoire à l'exécution. Une classe n'est donc qu'une collection de code qui modélise sous forme logicielle le comportement d'objets.

Le concept d'héritage de classe est respecté puisqu'il est possible de dériver une nouvelle classe d'objets à partir d'une classe originale dont on conserve les fonctionnalités en les étendant.

Par rapport au langage C++, Java apporte de nombreuses différences : pas de structures et d'unions, pas de *#define*, pas de pointeurs, pas d'héritage multiple (bien que Java permet à une classe d'implanter plus d'une interface), pas de surcharge d'opérateurs.

On pourrait voir Java comme un C++ allégé de sa complexité formelle.

Le travail effectué

L'objectif principal du stage est la réalisation d'un outil permettant

❶ la visualisation de grosses ontologies

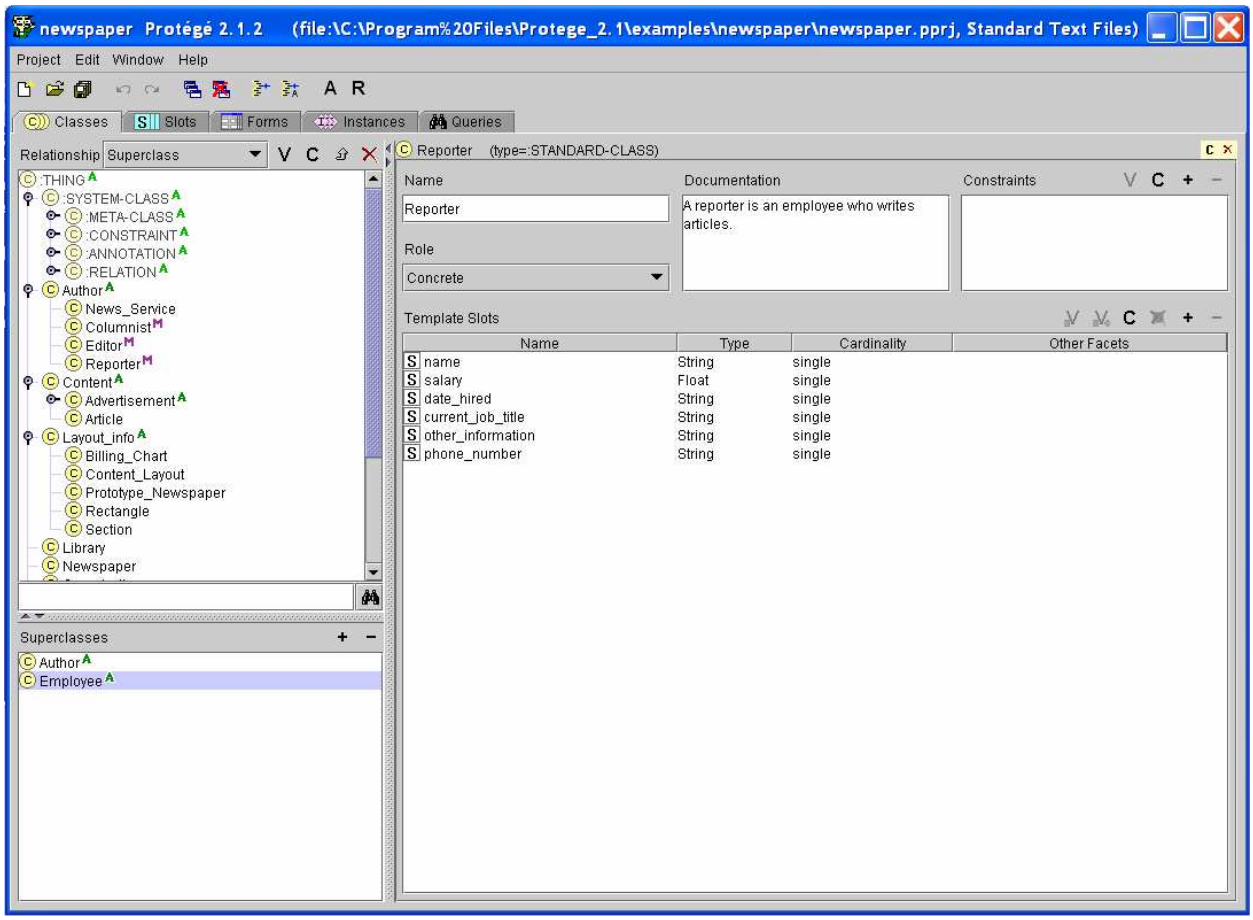
❷ le calcul et l'affichage des pouvoirs (résultats des mesures d'adéquation des ontologies à un corpus de documents) de celles-ci

❸ et enfin la possibilité de comparer deux ontologies simultanément (ce qui se traduit par l'affichage simultané de deux ontologies).

Analyse de l'existant

J'ai d'abord cherché des outils de visualisation d'ontologies existants capables de réaliser ou offrant la possibilité d'implémenter les trois objectifs cités précédemment.

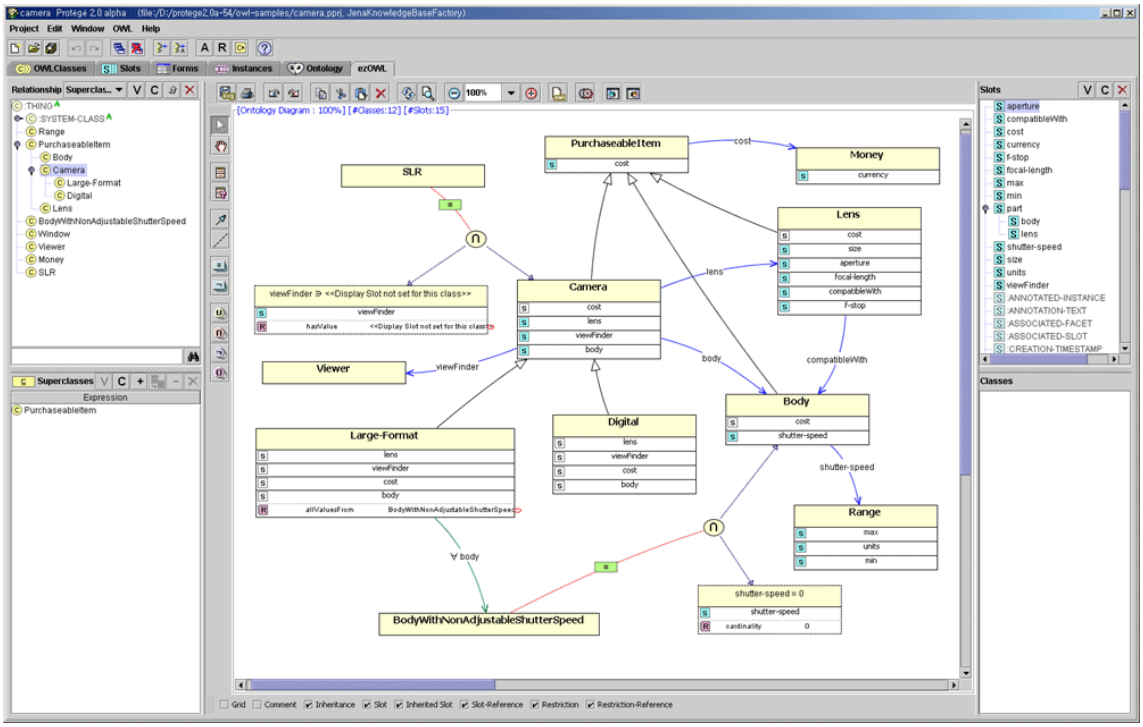
Il existe un logiciel développé par un groupe de l'Université de Médecine de Stanford (Etats-Unis), appelé *Protégé-2000*, dont le but principal est l'édition d'ontologies du domaine (protege.stanford.edu).



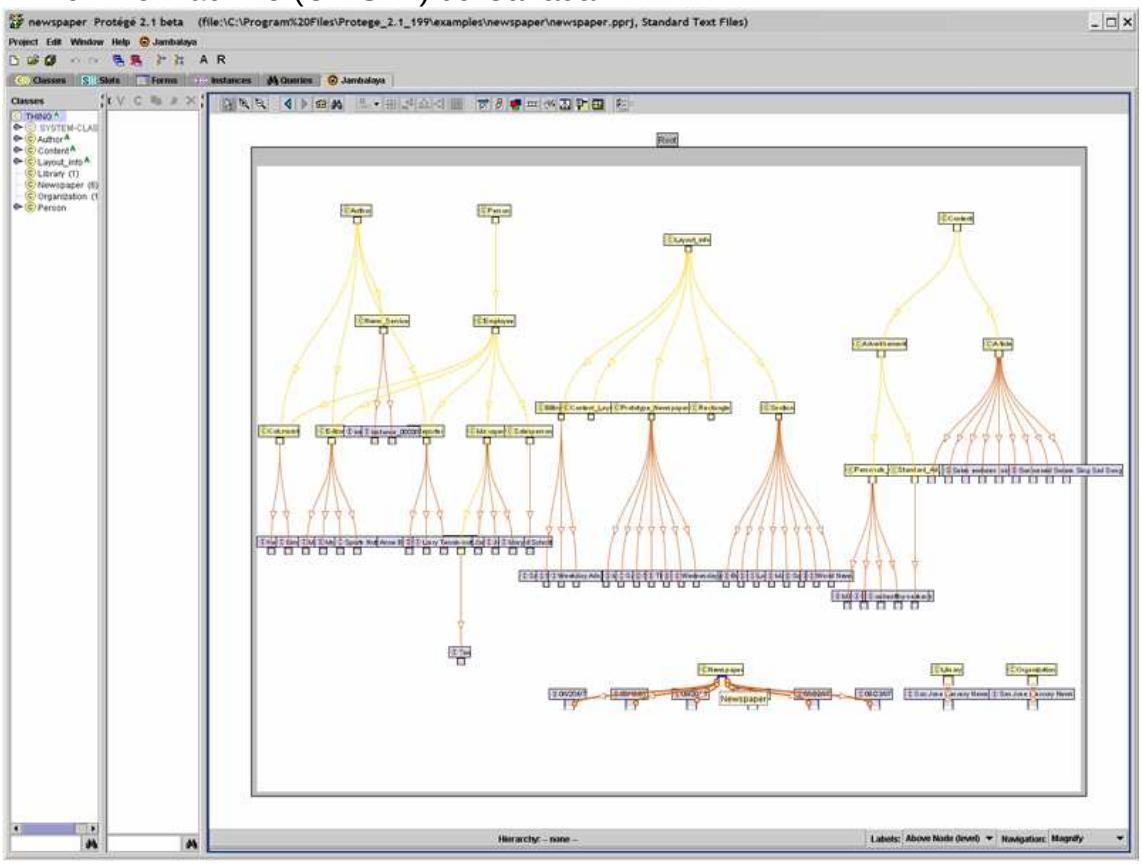
Telle quelle, cette application ne permet pas la visualisation d'ontologies. Par contre, grâce à l'ajout de plug-ins développés par d'autres groupes (*Protégé-2000* étant un projet open source), qui vont ajouter des onglets à l'application principale, on peut afficher des ontologies.

Il existe à ce jour cinq plug-ins de visualisation développés :

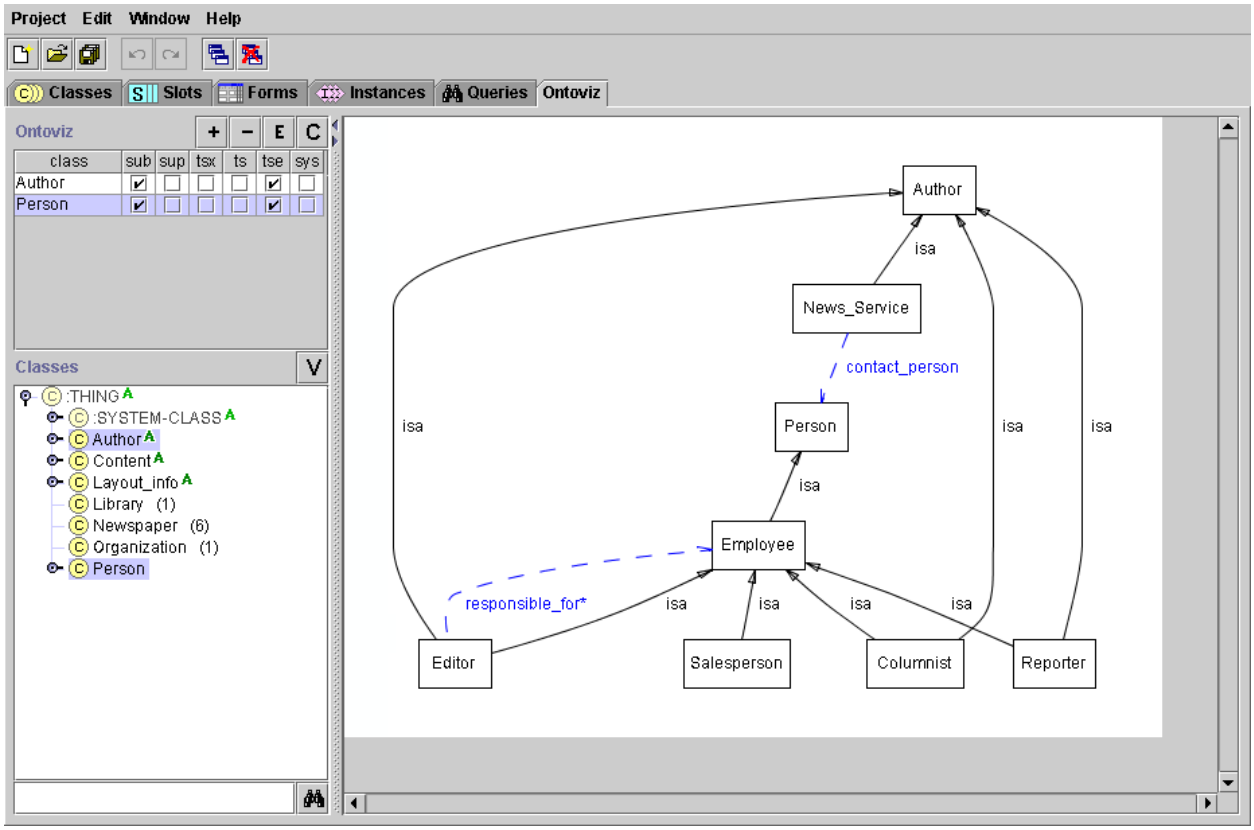
- ezOWL, réalisé par un groupe de l'*Institut de Recherche en Electronique et Télécommunications* de Corée.



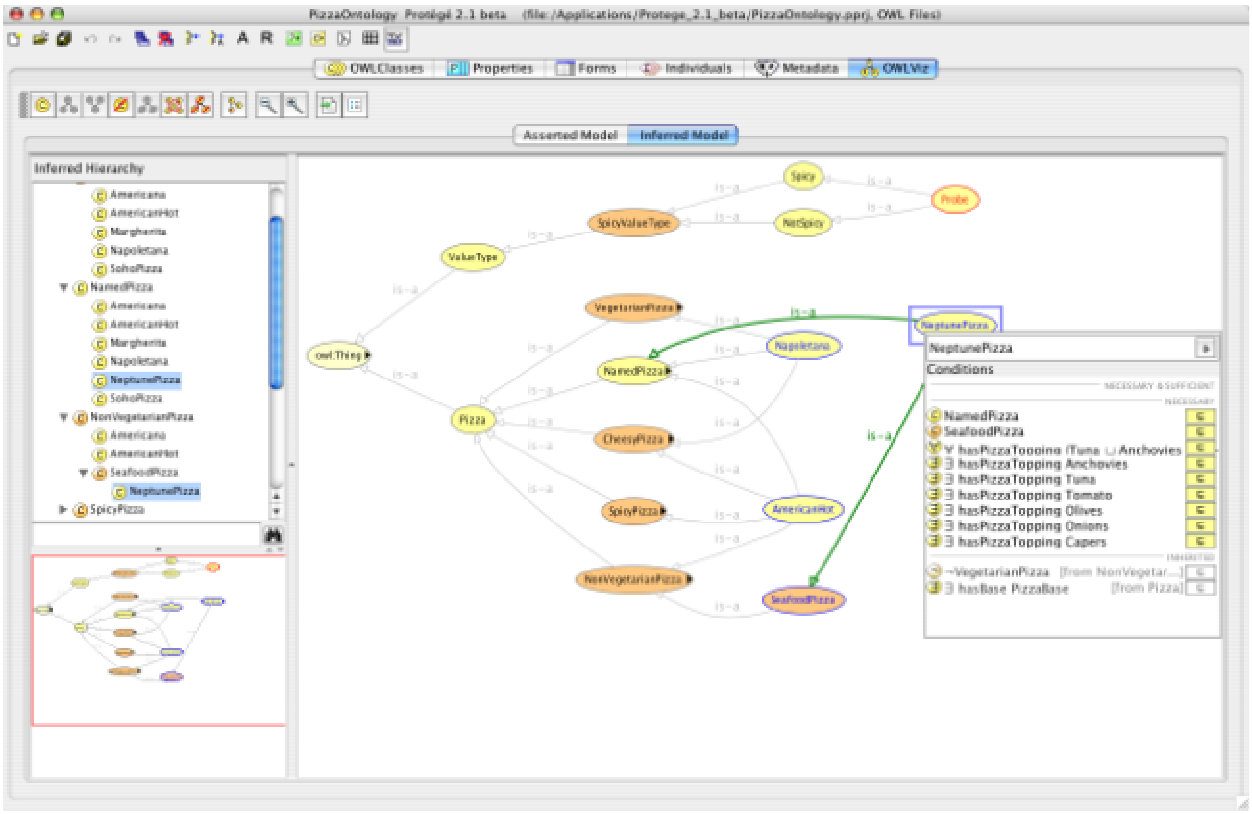
- Jambalaya, conçu par le *Laboratoire d'Ingénierie de Logiciels et d'Interactions Homme Machine* (CHISEL) du Canada.



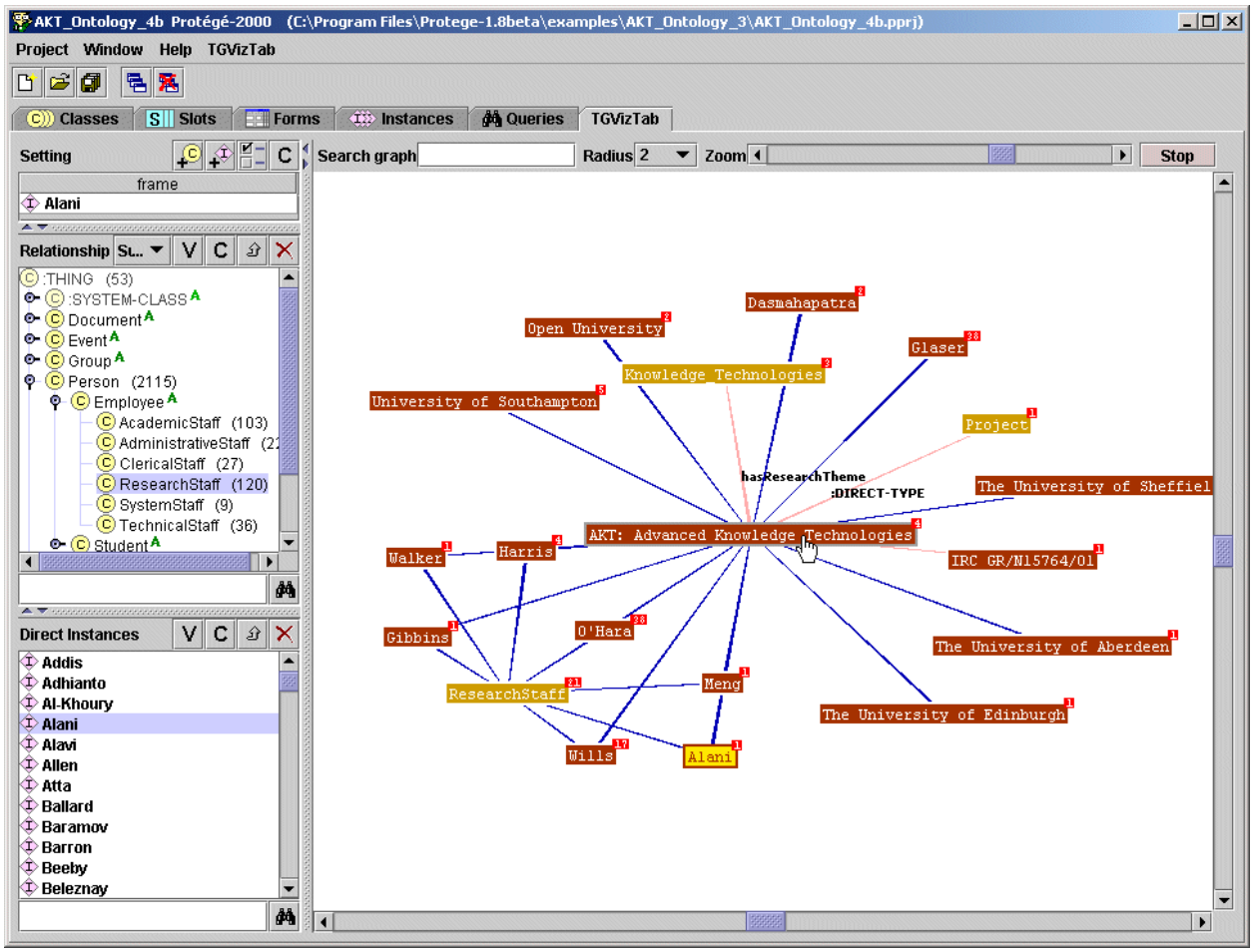
- ☆ OntoViz, développé par le *Centre de Recherche Allemand d'Intelligence Artificielle*.



- ☆ OWLViz, réalisé par un membre de l'*Université de Manchester* au Royaume-Uni.



- ☆ TGViz, conçu par un membre du *Département d'Intelligence, d'Agents, de Multimédia, d'Electronique et de Science Informatique de l'Université de Southampton* au Royaume-Uni.



J'ai donc décidé de comparer ces divers plug-ins et de faire ressortir leurs avantages et leurs inconvénients vis-à-vis des objectifs demandés de l'application finale.

Il en résulte le tableau de comparaison suivant :

	ezOWL	Jambalaya
Affichage d'un élément	Par double-clic. Se positionne sur l'élément correspondant dans la fenêtre.	On choisit les classes à afficher dans une liste présentée dans une fenêtre à part.
Zoom	Oui. Si on recule trop, on ne voit plus rien.	Oui. On zoome sur une classe et celle-ci s'affiche en prenant tout l'écran. Il n'est pas possible de tempérer le zoom afin de ne visualiser que certaines classes.
Automatisme	Les flèches entre chaque élément sont gérées toutes seules, mais avec possibilité de les déplacer manuellement.	Par défaut, les classes sélectionnées sont affichées, et toutes les propriétés entre celles-ci sont affichées (ce qui en cas de nombreuses propriétés devient vite illisible), avec possibilité de les masquer.
Affichage d'un graphe	Affichage pyramidal, hiérarchique.	Au choix, en étoile, pyramidal ou plat, avec possibilité de changer à volonté la disposition une fois l'ontologie affichée.
Préférences	Les couleurs des éléments peuvent être modifiées. Le contenu des éléments peut être modifié.	Oui, mais non intéressantes.
Autres	Simple d'utilisation. Très bon affichage.	Rapide, convivial et agréable, mais testé uniquement sur de très petites ontologies.

	OntoViz	OWLViz	TGViz
Affichage d'un élément	Peu intuitif. Par double-clic, on affiche en rouge l'élément correspondant dans la fenêtre.	Simple. Clic sur le bouton correspondant dans la barre de menu. Se positionne sur l'élément correspondant dans la fenêtre.	Assez compliqué. Pas intuitif.
Zoom	Oui. Si on recule trop, on ne voit plus rien.	Oui. Non testé.	Oui. Le fait de reculer ne modifie pas la taille des éléments, ils sont donc toujours parfaitement visibles.
Automatisme		Non testé.	L'affichage d'un graphe se fait automatiquement au centre. L'emplacement des éléments dépendants d'un autre est automatique, mais il n'est pas possible de les mettre comme on veut.
Affichage d'un graphe	Affichage hiérarchique. On ne peut pas bouger les éléments dans le graphe.	Affichage hiérarchique. Possibilité d'afficher ou pas les descendants d'une classe ou ses parents.	Affichage en étoile.
Préférences		Non testé.	Tout ce qui concerne l'affichage des éléments (couleur, police...) peut être modifié. Choix des liens à afficher.
Autres	Nécessite l'installation de GraphViz.	Nécessite l'installation de GraphViz. Il faut Protégé 2.1 (bêta). Semble être le plus simple.	Assez compliqué a utiliser, car peu intuitif. Excellent affichage, très agréable.

Au vu de ces résultats, aucune plug-ins ne correspond parfaitement aux besoins exigés. De plus, Protégé-2000 ne permet pas l'affichage simultané de deux ontologies,

et le développement d'un plug-ins pour Protégé-2000 ne permet pas l'affichage d'une seconde ontologie et exige de s'adapter à l'application existante, avec les contraintes et les problèmes que cela engendre.

Il a donc été convenu avec Nathalie Hernandez, et après accord de Josiane Mothe, de développer une nouvelle application qui répondrait parfaitement aux besoins exprimés.

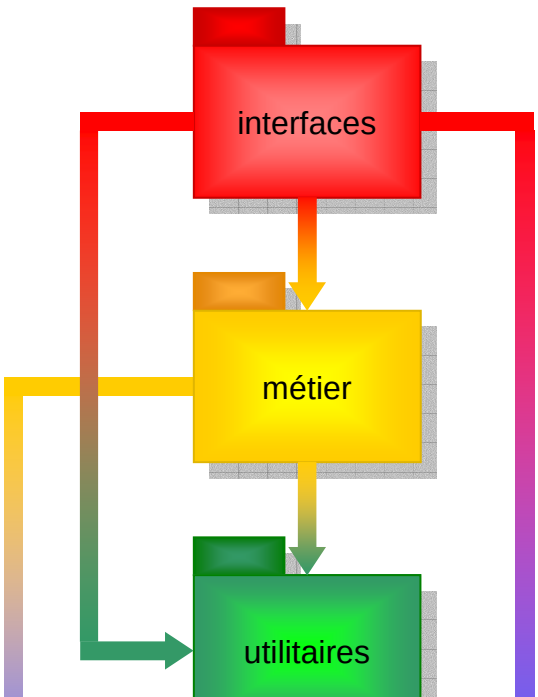
Spécifications de la nouvelle application : élégie

Après analyse des besoins et des contraintes exigées, j'ai réalisé une spécification de la future application et identifié son architecture.

La structure globale de l'application est une organisation du type Entity/Control/Bundary. Le principal but de cette architecture est le découpage entre l'ontologie et la présentation de celle-ci dans l'application. De plus, j'ai essayé de me focaliser sur une extension aisée, c'est-à-dire une maintenance facilitée, des traitements effectués sur l'ontologie et des interactions possibles entre l'utilisateur et l'application.

Organisation en couches

Il existe quatre couches : *interfaces*, *métier*, *utilitaires* et *autres*, dont les dépendances sont expliquées dans la figure suivante.



- Couche *interfaces*

  Cette couche a pour but la présentation des données à l'utilisateur. Elle lui permet aussi d'agir sur le modèle sous-jacent. Grâce à elle, il peut éditer une ontologie et lancer les opérations disponibles dans la couche *métier*. La couche interfaces fait appel aux classes définissant l'*ihm* du logiciel *élégie*.

- Couche *métier*

  Cette couche regroupe les classes métiers de l'application (*entité*) et l'implantation des règles de gestion spécifiques (*contrôle*). C'est cette couche qui est la plus liée aux cas d'utilisation du projet.

- Couche *utilitaires*

  Cette couche regroupe les fonctionnalités annexes au logiciel et qui sont directement liées au *métier*, par exemple les préférences, la gestion de l'internationalisation ou la gestion de l'affichage des graphes.

- Couche *autres*

  Cette couche regroupe les composants ne répondant pas aux spécifications des trois couches précédentes, et principalement des classes statiques permettant l'accès de données directement dans n'importe quelle classe de l'application.

Paquetages

couche	paquetages
interfaces	ihm
métier	contrôle, entité
utilitaires	graphes, messages, mesures, parcourir, préférences
autres	*aucun*

- Paquetage *ihm*

  Ce paquetage regroupe les classes relatives à l'ihm de l'application.

  ➔ BarreDeMenu

  Elle permet la création de la barre de menu principale de l'application. Pour ajouter un menu, il suffit de rajouter une action que l'on récupère dans le contexte à partir de sa clé. L'action doit se trouver dans le paquetage *entité*, et doit être répertoriée dans le contexte avec une clé dans le programme main de l'application.

  ➔ BarreOutil

  Elle permet la création de la barre outil principale de l'application. Pour ajouter un bouton, il suffit de rajouter une action que l'on récupère dans le contexte à partir de sa clé. L'action doit se trouver dans le paquetage *présentation*, et doit être répertoriée dans le contexte avec une clé dans le programme main de l'application.

  ➔ BarreOutilFenetre

  Elle permet la création des barres outils des deux fenêtres de l'application. Pour ajouter un bouton, il suffit de rajouter une action que l'on récupère dans le contexte à partir de sa clé. L'action doit se trouver dans le paquetage *présentation*, et doit être répertoriée dans le contexte avec une clé dans le programme main de l'application.

  ➔ FenChoixAppli

  Elle permet de choisir l'application que l'on veut lancer. Si l'on veut ajouter un autre choix, il faut créer un nouveau bouton radio auquel on associe le nouveau choix. Ce fichier contient également l'écouteur associé qui, lors de la validation du choix, va enregistrer dans le contexte le choix de l'application à lancer. Ce choix va être ensuite consulté par la fenêtre principale pour construire la fenêtre adéquate de l'application en fonction du choix.

  ➔ FenChoixClasseAffichage

  Dans le cas de grosses ontologies, elle permet de choisir les classes du premier niveau (au maximum cinq) à afficher parmi des listes déroulantes.

  ➔ FenChoixNiveauAffichage

  Dans le cas de grosses ontologies, elle permet de choisir le nombre de niveau maximal à afficher à partir du premier.

  ➔ FenDetailPouvoirs

  Après avoir calculer les mesures d'adéquation d'une ontologie (pouvoirs), cette fenêtre est affichée pour donner le pouvoir de l'ontologie ainsi que le pouvoir exact de la classe sélectionnée dans le graphe.

  ➔ FenListeLiens

  Elle permet l'affichage à l'intérieur d'une fenêtre interne à chacun des graphes la liste des liens autres que ceux d'héritages entre les différentes entités du graphe, c'est-à-dire toutes les propriétés objets. Ce fichier contient également

l'écouteur associé, qui lors d'un clic sur un lien dans la liste, permet l'affichage ou non du lien concerné.

- FenOuvrirFichier
  Elle crée la fenêtre permettant de choisir une ontologie, soit à partir de son adresse, soit en parcourant le système de fichier. Le chemin de l'ontologie est ensuite répertorié après validation dans le contexte.
- FenPrincipale
  Elle crée la fenêtre principale de l'application en fonction du choix de l'utilisateur sur l'application à lancer. Ce choix est consulté dans le contexte.
- FenProgression
  Elle affiche une barre de progression pour suivre le chargement d'une ontologie lors de son affichage.
- IconManager
  Manager pour récupérer des icônes.
- InterfaceFenPrincipale
  Interface de la fenêtre principale.

- Paquetage contrôle
  Ce paquetage regroupe les différents écouteurs de l'application.
  - EcouteurFenPreferences
    Il s'agit de l'écouteur de la fenêtre permettant de changer les préférences de l'application. Il se charge notamment de gérer le choix des couleurs.
  - EcouteurMenuPopupInstance
    Il s'agit de l'écouteur des menus popup qui affichent lors d'un clic droit sur une classe d'un graphe les instances relatives à cette classe.
  - EcouteurSourisGraphe
    Il s'agit de l'écouteur de la souris sur le graphe. Il gère notamment le clic droit sur les classes en affichant dans un menu popup les instances de la classe concernée, ainsi qu'après avoir calculer les pouvoirs d'une ontologie le pouvoir de la classe sélectionné dans le graphe.

- Paquetage *entité*
  Ce paquetage concerne les différentes actions possibles de l'application.
  - Actions
    Classe générale du paquetage. Elle définit les trois constructeurs utiles permettant la création d'actions auxquelles sont associées un label (utilisé dans la barre de menu ainsi que dans les info-bulles des barres outils), une icône (affichée dans la barre de menu et dans les barres outils) et éventuellement un raccourci clavier.
  - CalculerPouvoirAction
    Action destinée à calculer les mesures d'adéquation d'une ontologie, c'est-à-dire son pouvoir.
  - ChargerCorpusAction
    Action destinée à charger un corpus de document dans le but de calculer plus tard l'adéquation des ontologies sur ce corpus.
  - FermerFichierAction
    Action destinée à fermer une ontologie. Elle fait appel à la procédure *fermerFenetre* de *FenPrincipale* qui gère de fermer la fenêtre et de réinitialiser tous les champs utilisés dans le contexte.

- OuvrirFichierAction
  Action destinée à ouvrir une ontologie. Elle affiche la fenêtre de saisie du chemin de l'ontologie à ouvrir, puis récupère l'URI dans le contexte pour savoir si elle a été saisie, et en fonction, affiche le graphe associé, puis crée un écouteur et l'associe au graphe.
- PreferencesAction
  Action destinée à afficher la fenêtre de modification et d'enregistrement des préférences de l'application, notamment le choix de la langue, du répertoire d'ouverture des fichiers, et des différentes couleurs utilisées lors de l'affichage d'un graphe et des calculs des pouvoirs.
- ProjectManagementAction
  Classe dont toutes les actions du paquetage héritent. Elle même héritent de Actions.
- QuitterAction
  Action destinée à gérer la fermeture de l'application (affichage des différentes fenêtres de confirmation de fermeture...).
- ZoomEntrantAction
  Action destinée à gérer le zoom entrant sur un graphe.
- ZoomSortantAction
  Action destinée à gérer le zoom sortant sur un graphe.

- Paquetage *graphes*

Ce paquetage concerne la gestion de l'affichage des graphes représentant les ontologies.

- AffichageGraphe
  Cette classe affiche le graphe associé à une ontologie. Pour cela, elle récupère d'abord l'URI de l'ontologie dans le contexte, puis va créer les classes, les arranger correctement, puis créer tous les liens d'héritage et les insérer dans le graphe, et enfin créer tous les autres liens et les associer à une fenêtre interne FenListeLiens. Elle affiche ensuite le graphe au bon endroit et le répertorie dans le contexte.

- Paquetage *messages*

Ce paquetage concerne la gestion de l'affichage des fenêtres de messages.

- ErrorManager
  Cette classe gère les fenêtres de message d'erreur. Pour afficher un message, il faut passer en paramètre les clés correspondant aux textes répertoriés dans les fichiers de langues.

- Paquetage *mesures*

Ce paquetage concerne le calcul des mesures d'adéquation des ontologies.

- Pouvoir
  Cette classe se charge de calculer les mesures d'adéquation (pouvoirs) d'une ontologie vis-à-vis d'un corpus de documents.

- StatOnto
  Cette classe enregistre les résultats du calcul des pouvoirs. Un premier attribut répertorie le pouvoir général de l'ontologie, un second enregistre sous la forme d'une HashMap les pouvoirs de chaque classe.

➔ Subsumeur

Cette classe est utilisée lors du calcul des pouvoirs d'une ontologie.

- Paquetage *parcourir*

Ce paquetage concerne les filtres utilisés lors du parcours des systèmes de fichier au travers d'explorateurs.

➔ Filtre

Il s'agit d'un filtre qui peut être utilisé pour les explorateurs de fichiers.

- Paquetage *préférences*

Ce paquetage concerne la gestion des préférences et de l'internationalisation

➔ PreferencesManager

Il s'agit d'un gestionnaire permettant l'enregistrement des préférences de l'utilisateur concernant l'application. Pour chaque préférence, on crée deux méthodes : l'une pour récupérer la préférence, l'autre pour l'enregistrer. A chaque préférence, on associe une clé.

➔ ResourceManager

Il s'agit du gestionnaire de langue de l'application. Dans toute l'application, tous les textes affichés passe par ce gestionnaire qui permet ainsi l'internationalisation de l'application. Ainsi, pour afficher un texte, il faut le récupérer via ce gestionnaire grâce à la clé associée au texte. Les différents textes et clés sont enregistrés dans les fichiers de langue **elegie_XX.properties**, où XX représente le code international de la langue correspondant (**fr** pour français, **en** pour anglais...).

- Couche *autres*

➔ ContextElegie

Le contexte permet la sauvegarde de tout un tas de données utiles pour l'application. Il possède pour attributs toutes les données à sauvegarder, et pour méthodes les fonctions permettant de récupérer ou modifier ces données. Le contexte facilite ainsi l'accès à une donnée dans toutes les classes de l'application, et ce, sans passage de paramètres.

- Classe principale

➔ MainElegie

C'est la classe principale de l'application. Elle fournit d'abord au gestionnaire de langue le chemin du fichier de langue, puis elle affiche la fenêtre du choix de l'application. Ensuite, elle enregistre dans le contexte toutes les actions qui vont être utilisées dans l'application, et enfin, elle crée et affiche la fenêtre principale.

Affichage d'un graphe

Pour la visualisation d'une ontologie, j'ai choisi un affichage hiérarchique selon les liens d'héritages. Ainsi, sont considérées au niveau 0, et donc affichées tout en haut

du graphe, les classes de l'ontologie qui n'ont pas de classes mères. Au niveau suivant, le premier niveau, sont affichées toutes les classes qui ont pour classes mères les classes du niveau 0, et uniquement celle-là. Et ainsi de suite, pour qu'une classe puisse être affichée, il faut que celle-ci ait au moins une de ses classes mères affichées.

Dès lors que j'ai défini cet affichage, j'ai réalisé la programmation. Tout d'abord, je lance en parallèle la barre de progression et le chargement de l'ontologie avec des threads. Le chargement correspond à récupérer l'URI de l'ontologie saisie par l'utilisateur dans le contexte, puis à la parser grâce à une API développée par un groupe de l'Université de Manchester dans le cadre de leur projet WonderWeb, OWLAPI, qui permet de manipuler en java une ontologie au format OWL. Cette API récupère toutes les classes avec leurs caractéristiques, ainsi que toutes les propriétés objets. Dès que l'ontologie a été parsée et récupérée, on range les classes de la manière décrite précédemment par niveau, puis on vérifie le nombre total de classes, et si celui-ci est trop élevé, on affiche une fenêtre permettant à l'utilisateur de choisir le nombre maximal de niveau à afficher, puis parmi les classes du premier niveau, celles qu'il veut afficher (et donc en fonction de celles-ci les classes des niveaux suivants), au maximum cinq. Si le nombre de classes total n'est pas élevé, toutes vont être affichées.

On parcourt ensuite toutes les classes niveau par niveau. Toutes les classes du niveau 0 sont affichées, pour les classes du premier niveau, seules les classes choisies sont affichées, puis pour les classes des niveaux suivant, il faut que au moins une de leurs classes mères soit affichée. Si une classe peut-être affichée, on crée une cellule graphique avec JGraph, puis on l'insère dans le graphe. Pour l'instant, les cellules ne sont ordonnées que par niveau, c'est-à-dire sur l'axe vertical.

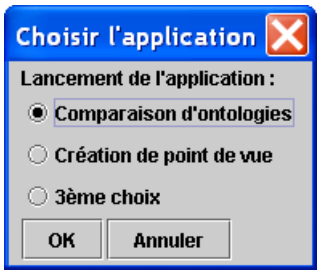
Une fois que toutes les cellules ont été crées et insérées dans le graphe, on ordonne les cellules sur chaque niveau. D'abord, on centre les cellules sur leur niveau, c'est-à-dire que en fonction de la longueur maximale des niveaux, on place les cellules par rapport au milieu de cette longueur maximale. Par ailleurs, sur un niveau, les cellules sont ordonnées par rapport à l'ordre des cellules précédentes. C'est-à-dire que l'on va mettre à gauche les cellules dont les classes mères sont à gauche, puis les cellules dont les classes mères sont à la droite des classes mères des cellules précédentes, et ainsi de suite. Maintenant, le graphe est ordonné.

Ensuite, on crée les arcs représentant les liens d'héritage (relations mère-fille). Seuls les arcs dont les deux classes source et cible sont affichées peuvent être affichés. Pour les propriétés objets, on ne crée que les arcs qui ont au moins une cellule (source ou cible) qui est affichée. Si les deux cellules sont affichées, on relie les deux cellules avec cet arc. Si une seule cellule est affichée, un arc partant de celle-ci est crée. Les arcs représentant les propriétés objets ne sont pas insérés. Ils ne le seront que quand l'utilisateur le voudra en cliquant sur les propriétés qu'il désire visualiser dans la fenêtre des propriétés objets insérée dans le graphe.

Fonctionnement de l'application

Au lancement de l'application, une fenêtre demande choisir l'application que l'on veut lancer parmi une liste de trois choix : comparaison d'ontologies, création d'un point

de vue (application réalisée par Alexandre Bès), et une troisième application qui n'a pas encore été réalisée :



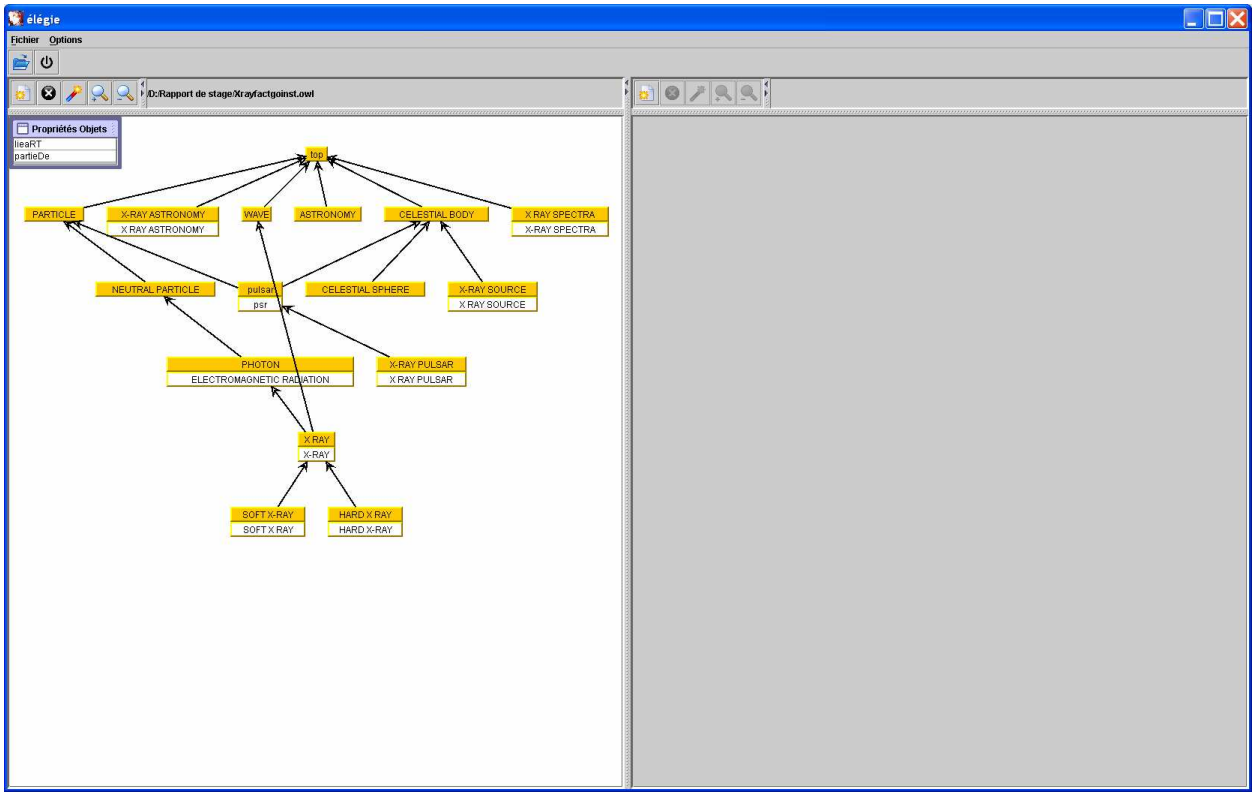
En fonction de l'application choisie, la fenêtre principale est lancée et adaptée au choix :

Celle-ci comprend une barre de menu permettant de charger un corpus, de quitter et de modifier les préférences, une barre outil pour charger un corpus et quitter, et deux bureaux identiques comprenant chacun une barre outil permettant d'ouvrir une ontologie, de la fermer, de calculer ses pouvoirs et de zoomer, et enfin la zone où sont affichés les graphes.

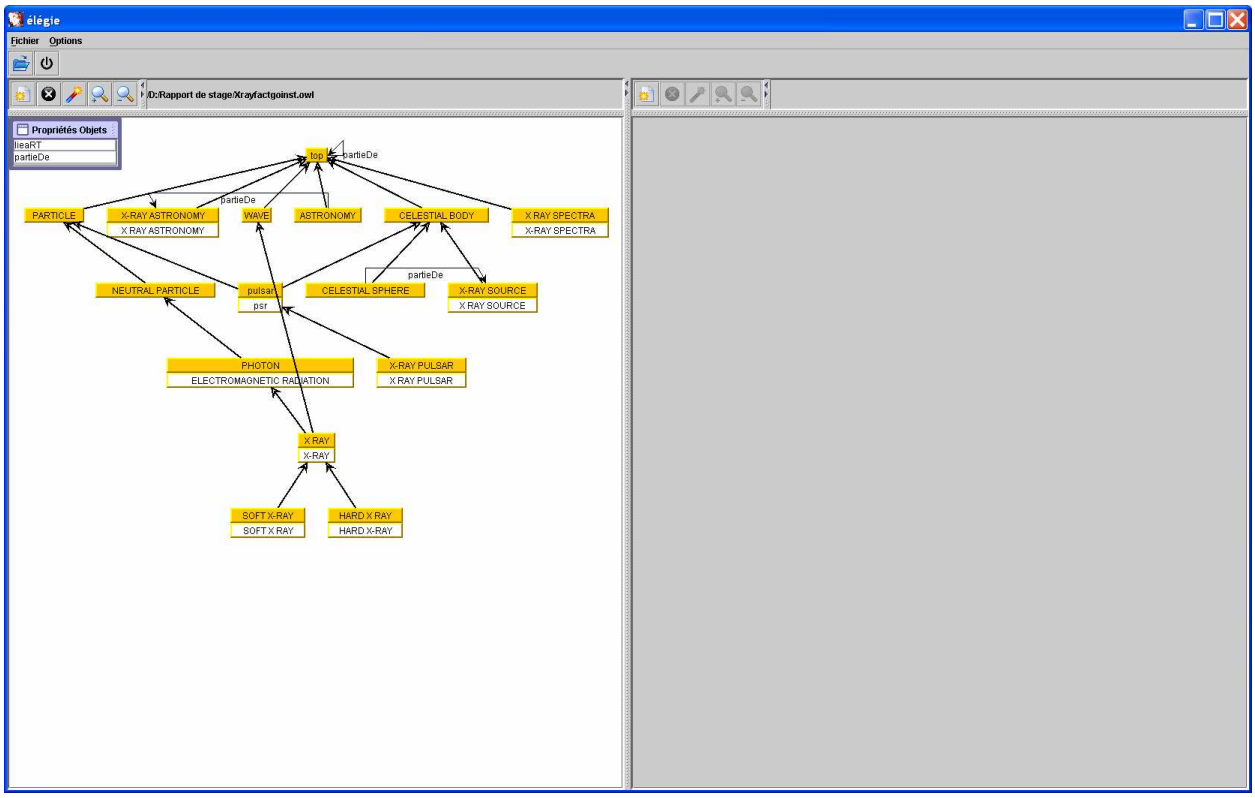
Pour ouvrir une ontologie, il suffit de cliquer sur l'icône correspondante dans la barre outil du côté où l'on veut afficher le graphe :

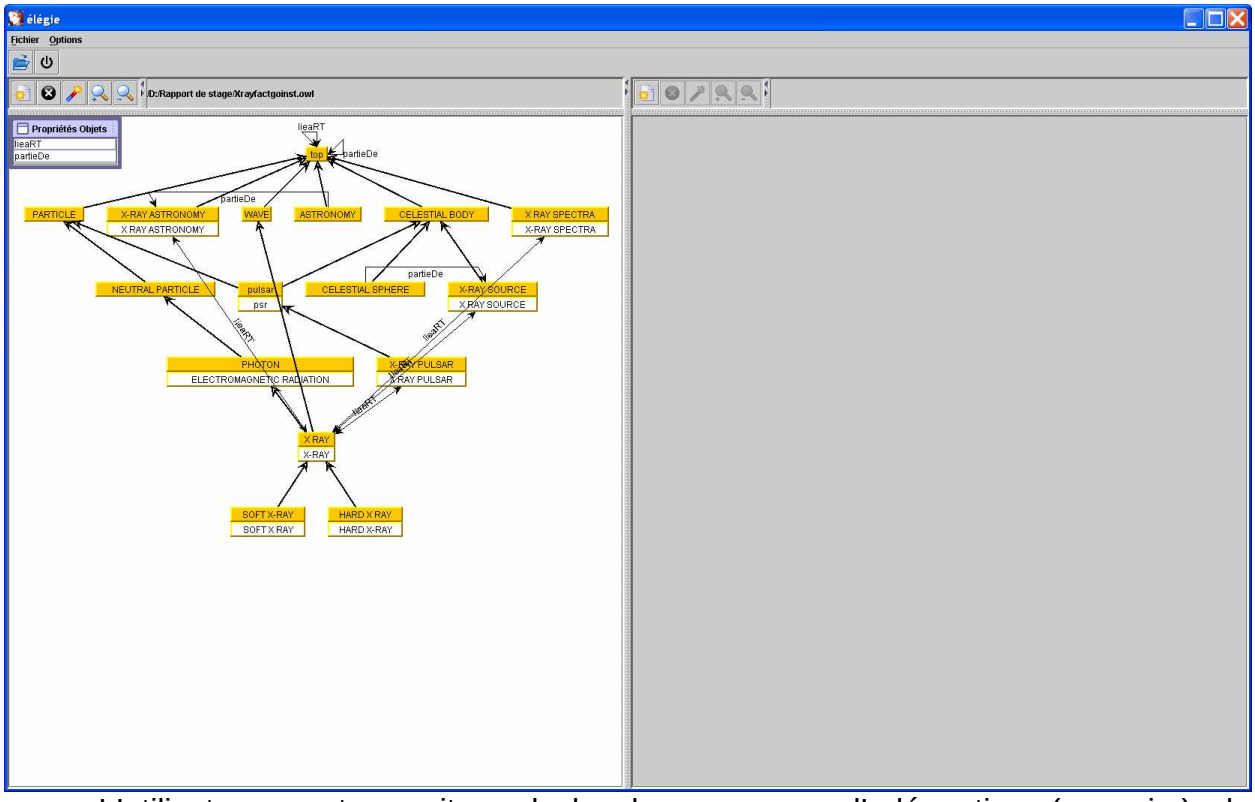
Une fenêtre s'affiche invitant l'utilisateur à saisir l'URI de l'ontologie à afficher. Il peut soit la saisir par le clavier, soit parcourir le système de fichiers pour la retrouver :

Une fois que l'utilisateur à cliquer sur OK, le graphe est affiché dans la bonne fenêtre :

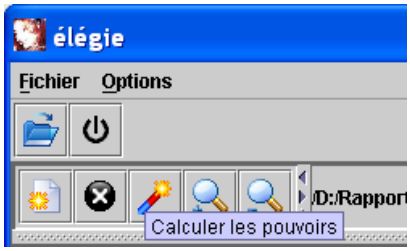


Les concepts de l'ontologie sont ordonnés par niveau. Seuls les liens d'héritage sont affichés. Mais dans une fenêtre interne se trouve la liste des autres liens possibles (correspondant aux propriétés objets). Il suffit de cliquer sur le lien que l'on désire afficher pour qu'il s'affiche. On peut de la même manière ne plus l'afficher :

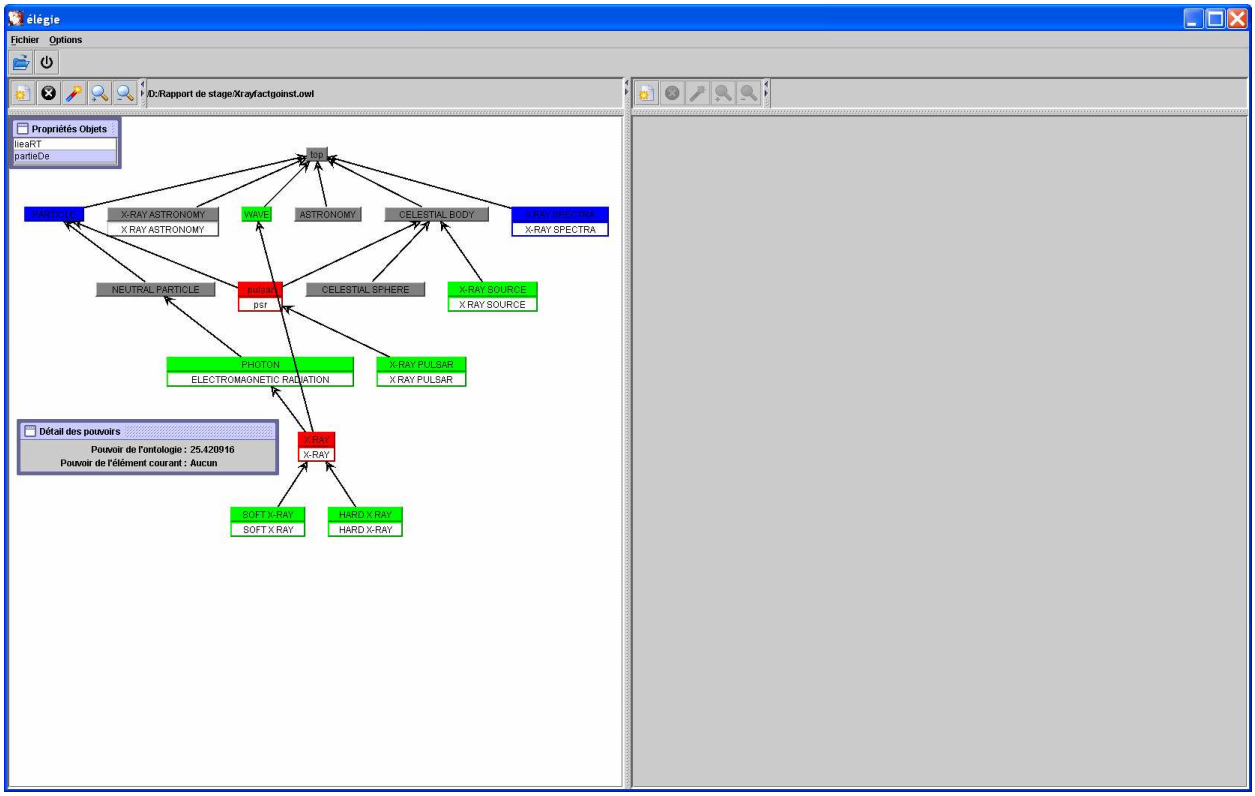




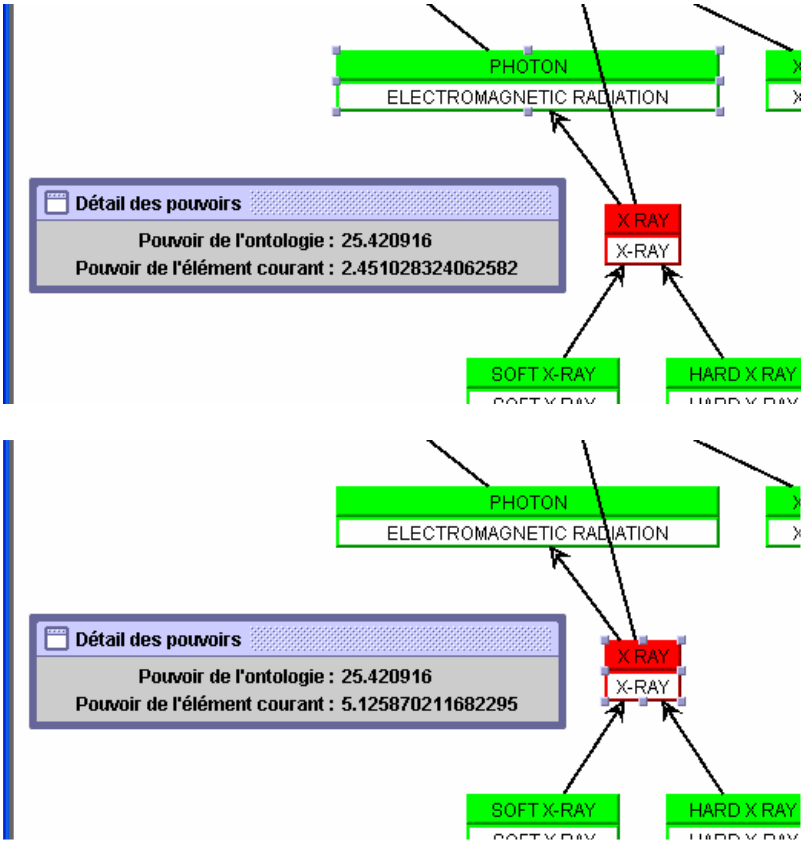
L'utilisateur peut ensuite calculer les mesures d'adéquation (pouvoirs) de l'ontologie. Il lui suffit de cliquer sur l'icône de la barre outil :



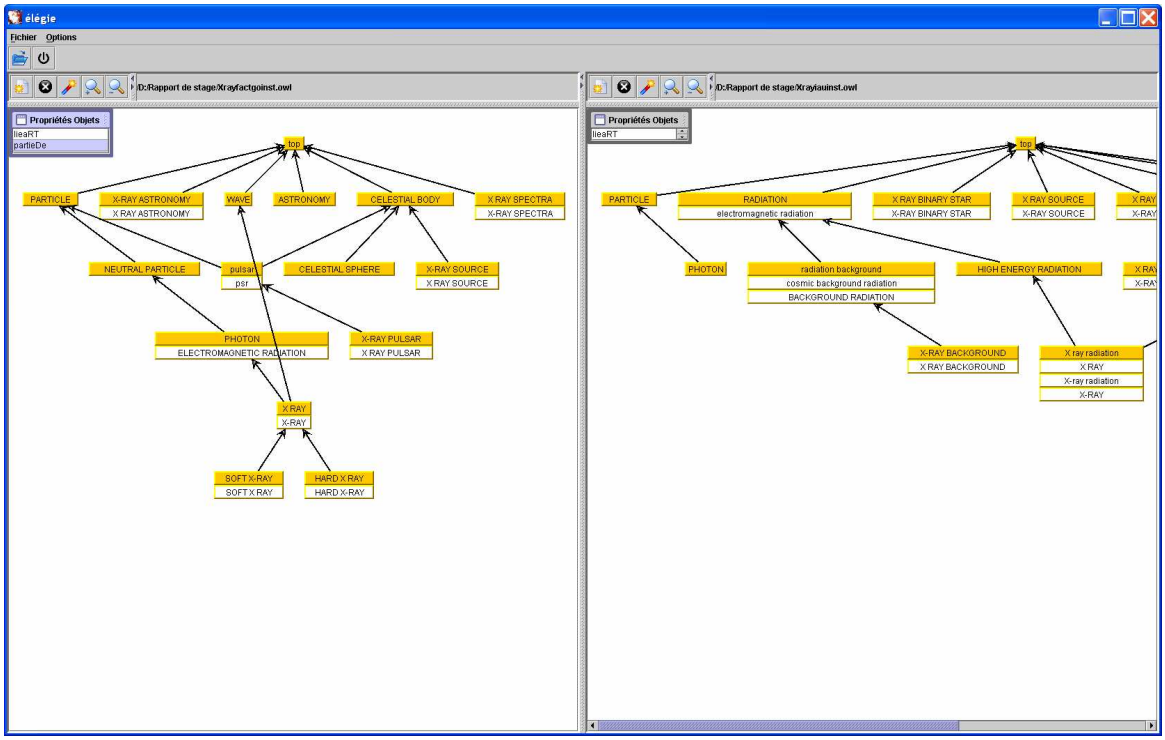
Toutes les cellules du graphe représentant les concepts de l'ontologie sont colorées en fonction de leur pouvoir. L'application récupère le plus fort et le plus faible pouvoir, et suivant la place du pouvoir du concept dans cet intervalle (premier 20%, deuxième 20%...), la couleur est différente. Les concepts qui n'ont pas de pouvoir ont une autre couleur :



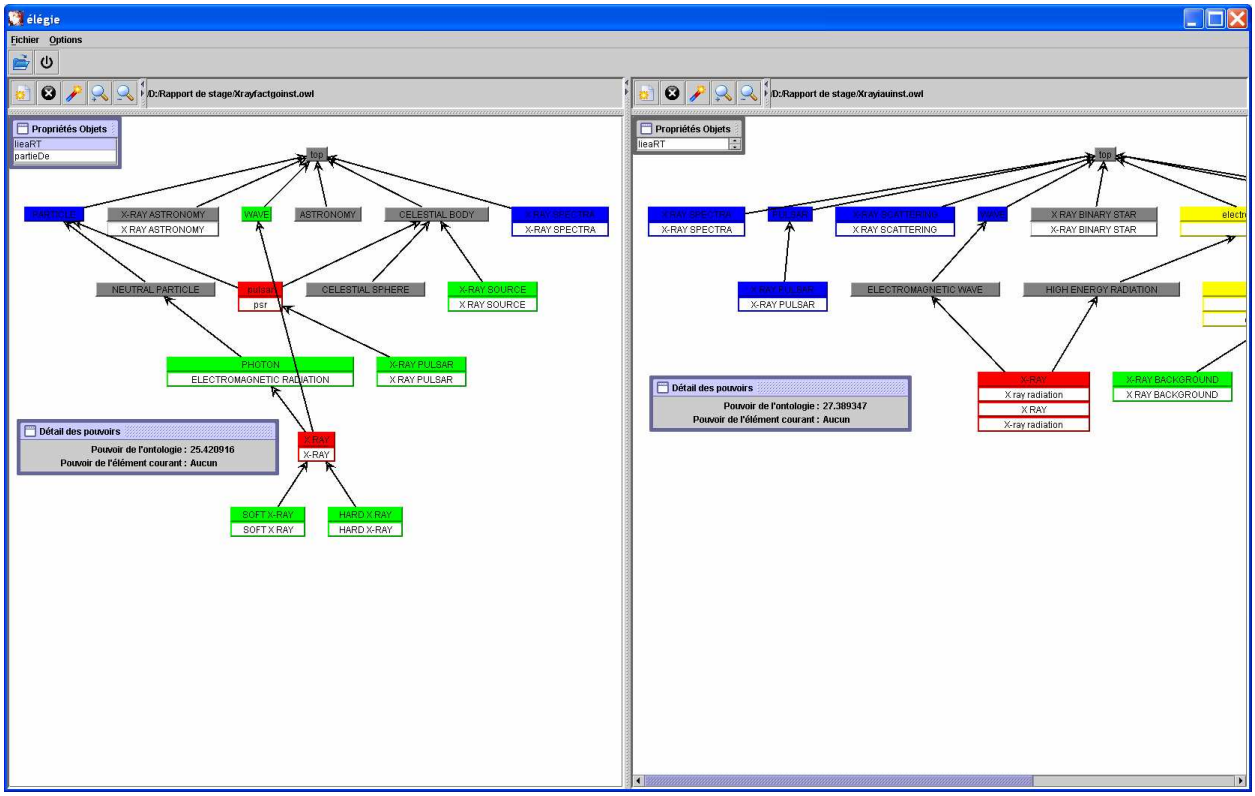
De plus, une fenêtre interne permet de visualiser le pouvoir total de l'ontologie, ainsi que le pouvoir exact du concept courant sélectionné :



L'utilisateur à la possibilité d'ouvrir une seconde ontologie simultanément de la même manière que la première :

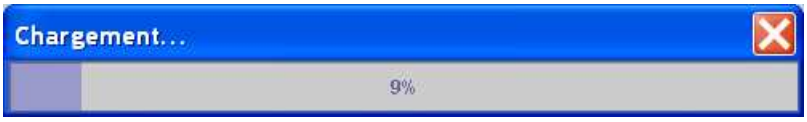


Et ensuite calculer les mesures d'adéquation de cette seconde ontologie afin de pouvoir comparer les deux ontologies ouvertes :

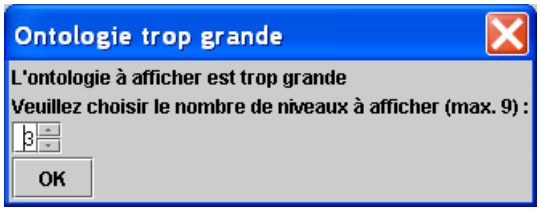


L'utilisateur à la possibilité de fermer l'ontologie qu'il désire en cliquant sur l'icône correspondante :

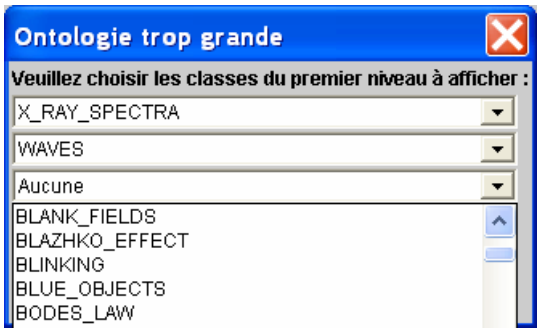
Dans le cas de grosses ontologies, d'abord, l'application affiche une barre de progression afin d'informer l'utilisateur sur l'avancement du chargement :

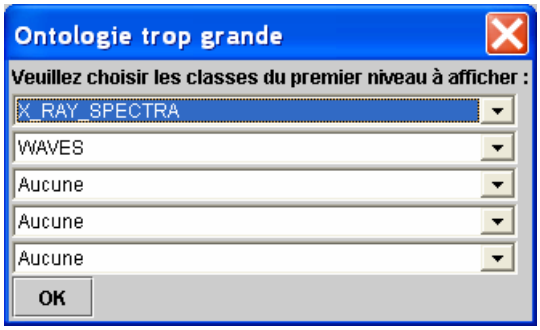


Et si l'ontologie est trop grande (trop de concepts définis), l'application propose d'abord de choisir le nombre de niveaux à afficher sur le nombre total de niveau :

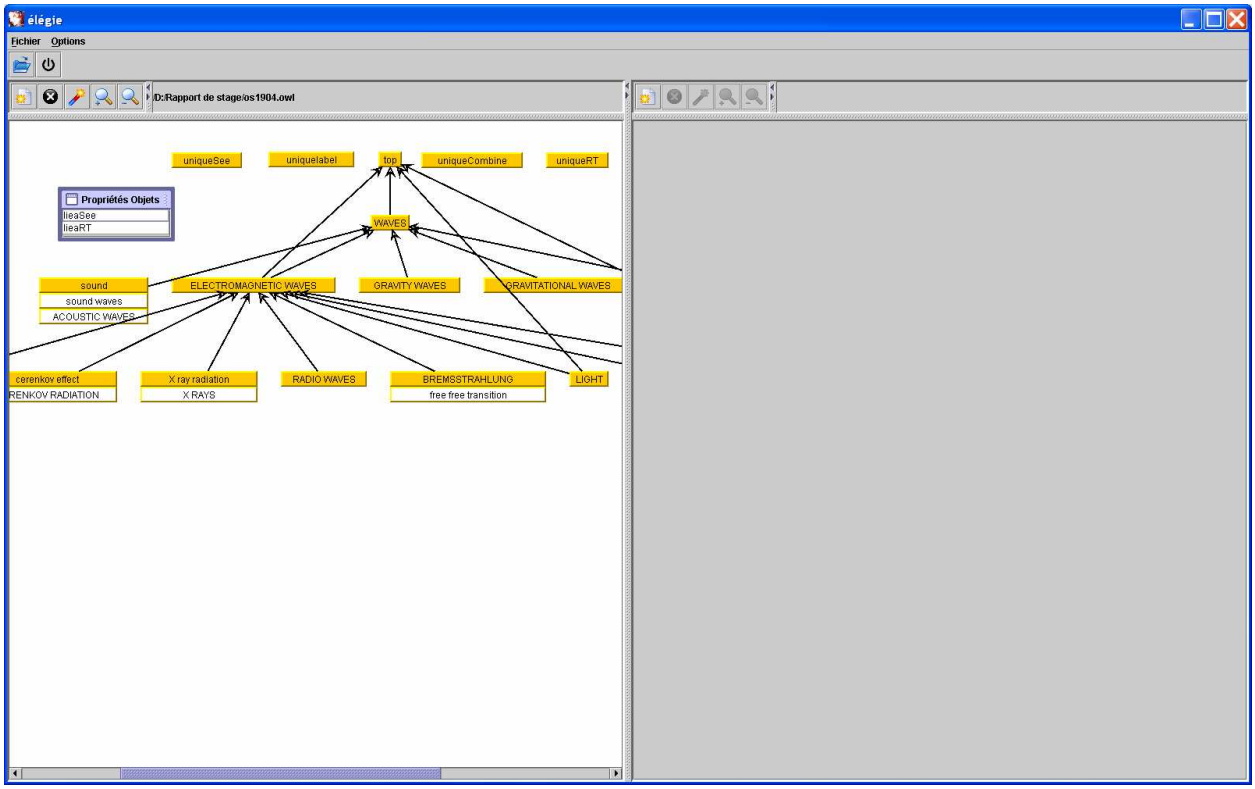


Puis de choisir les concepts du premier niveau à afficher parmi les listes proposées (cinq au maximum) :

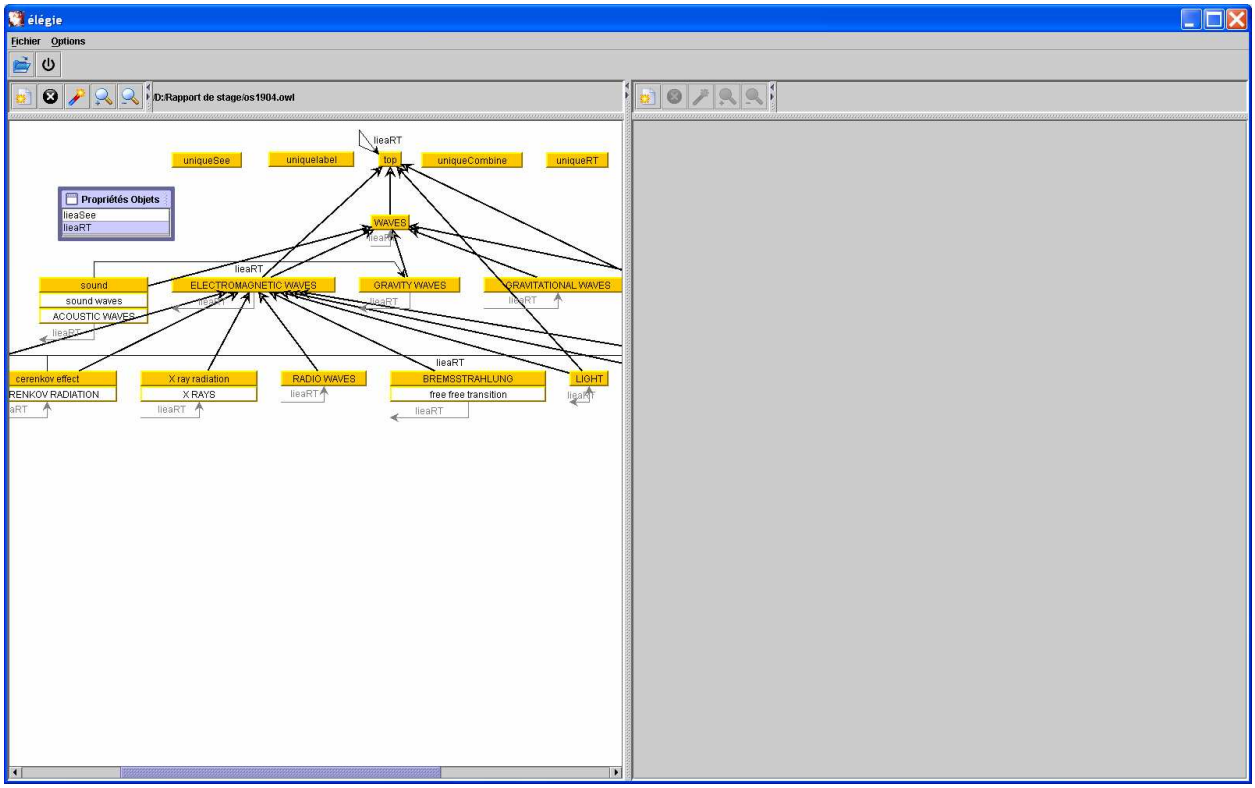




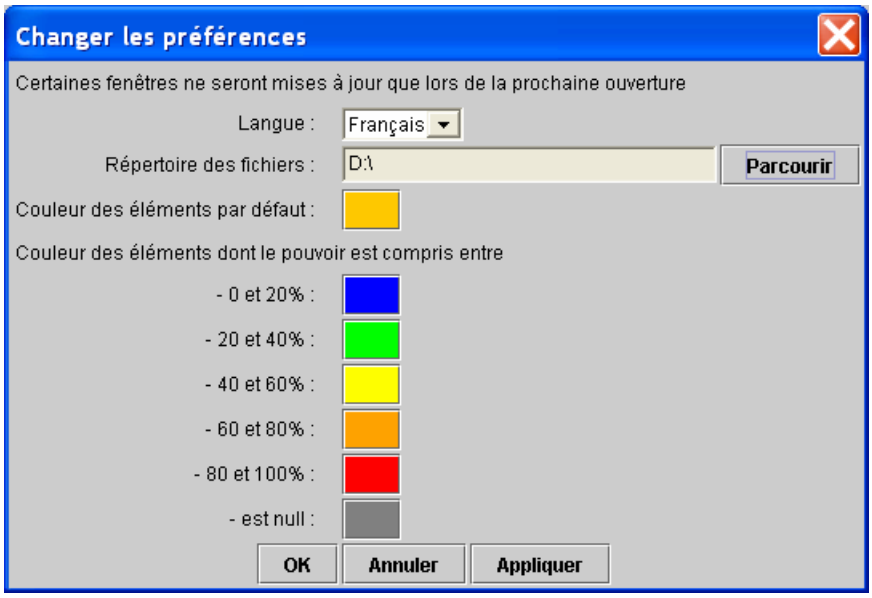
Et l'ontologie est affichée en fonction de ces critères :



Quand on choisit d'afficher les propriétés objets, les arcs dont un seul sommet (cellule représentant un concept) est affiché sont visualisés en dessous du concept correspondant et en gris :

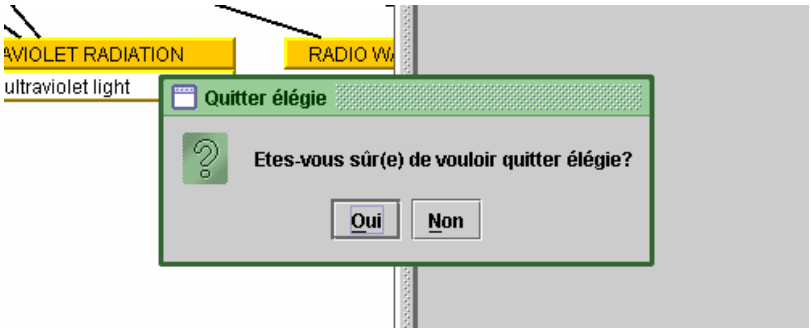


Par ailleurs, l'utilisateur peut modifier ses préférences, à savoir la langue, le répertoire d'ouverture des fichiers, et les différentes couleurs utilisées :



Et quand l'utilisateur désire quitter, il lui suffit de quitter en choisissant l'option dans la barre de menu ou en cliquant sur l'icône correspondante de la barre outil :

Une dernière fenêtre s'affiche pour demander confirmation de quitter l'application :



Les apports du stage

Au-delà du travail de conception et de programmation purs, j'ai eu l'occasion de participer aux divers événements propres au travail d'équipe en entreprise.

Réunion

Organisées régulièrement, les réunions m'ont apporté beaucoup d'expérience sur le plan relationnel. Elles amènent à cerner précisément les problèmes que l'on peut rencontrer au cours du travail effectué, confronter ses idées, affiner la conception et d'accroître ses connaissances.

C'est également un très bon exercice d'expression orale qui nous apprend à nous exprimer clairement et simplement, et également à écouter les autres.

Equipe

Tout au long de ce stage, j'ai évolué au sein d'une équipe, ce qui m'a amené à découvrir la vie de celle-ci au sein de l'entreprise, avec ses problèmes, son organisation, ses réunions… Une situation d'autant plus intéressante qu'elle m'a permis d'élargir mon champ de connaissance et de vision sur la vie en entreprise.

Conclusion

Ce stage représente ma troisième expérience en entreprise dans le secteur informatique. Et cette troisième expérience m'a été davantage enrichissante que les deux premières, et ce pour plusieurs raisons.

D'abord par le travail effectué, qui m'a permis de découvrir de nouveaux outils et d'élargir ma connaissance sur d'autres, mais également d'approfondir ce que j'ai pu apprendre à l'IUP et d'acquérir de nouvelles méthodes de travail.

Ensuite, l'insertion au sein d'une équipe et un bon encadrement m'a présenté et fait connaître la vie en entreprise, et ce de manière plutôt active et positive.

Enfin, une satisfaction plus personnelle d'avoir participé à la création d'un logiciel qui est déjà utilisé.

Au final, un stage que j'ai énormément apprécié, tant sur le plan professionnel, avec un sujet de stage fort intéressant, que sur le plan relationnel, grâce à l'accueil, l'entente et l'ambiance qui régnaient au sein de l'équipe.